Diplomarbeit

JWAM[1]-Komponenten zur Kooperationsunterstützung: Die Raumkomponente

Jörn Koch

Pfauenweg 42
22305 Hamburg

Oktober 2000

Erstbetreuung: Dr. Ingrid Wetzel
Zweitbetreuung: Prof. Dr. Arno Rolf

Fachbereich Informatik
Universität Hamburg
Vogt-Kölln-Straße 30
22527 Hamburg

[1] JWAM ist ein an der Universität Hamburg entwickeltes Java-Framework nach dem Werkzeug- und Material-Ansatz (WAM)

Erklärung:
Ich versichere hiermit, diese Arbeit selbständig und unter ausschließlicher Zuhilfename der in der Arbeit aufgeführten Hilfsmittel erstellt zu haben.

Hamburg, den 1. Oktober 2000

Jörn Koch
Pfauenweg 42
22305 Hamburg
1jkoch@gmx.de

Betreuung:

Dr. Ingrid Wetzel (Erstbetreuerin)

Prof. Dr. Arno Rolf (Zweitbetreuer)

Dr. Ingrid Wetzel

Arbeitsbereich Softwaretechnik
Fachbereich Informatik
Universität Hamburg
Vogt-Kölln-Str. 30
22527 Hamburg

Prof. Dr. Arno Rolf

Arbeitsbereich Angewandte und Sozialorientierte Informatik
Fachbereich Informatik
Universität Hamburg
Vogt-Kölln-Str. 30
22527 Hamburg

Inhaltsverzeichnis

0. Einleitung 6

0.1 Ziele der Arbeit 8
0.2 Aufbau der Arbeit 9
0.3 Konventionen 10
0.4 Danksagung 10

1. Einbettung der Arbeit 12

1.1 Der WAM-Ansatz 12
1.1.1 Softwareentwicklung als dokumentenbasierter Kommunikations- und Lernprozess 13
1.1.2 Softwareentwicklung als Modellierungsprozess 15
1.1.3 Das WAM-Leitbild mit seinen Entwurfsmetaphern 18

1.2 Kooperationsunterstützung und CSCW 23

1.3 Räumliche Konzepte in der CSCW-Diskussion und in Groupware-Systemen 27
1.3.1 Das locales framework und der Orbit-Mercury-Prototyp 28
1.3.2 Das TeamWave-System 31
1.3.3 Der POLITeam Awareness Client (POLIAwaC) 34
1.3.4 Vergleich der drei Groupware-Systeme 37
1.3.5 Re-Place-ing Space 39
1.3.6 Ort und Platz 41

1.4 Das JWAM-Framework 43
1.4.1 Frameworks 43
1.4.2 Komponenten 46
1.4.3 Die JWAM-Schichtenarchitektur und ihre Frameworks 48

1.5 COJAC – JWAM-Komponenten zur Kooperationsunterstützung 54

2. Fachliche Motivation für eine Raumkomponente 56

2.1 Zwei Szenarios aus der PR-Abteilung eines Buchverlags 57
2.1.1 Szenario "Mailing-Aktion" 59
2.1.2 Szenario "Anfrage annehmen" 60

2.2 Anforderungsszenarien und Lösungskonzepte 61
2.2.1 Koordination über Material und Platz 61
2.2.2 Gemeinsame Nutzung knapper Ressourcen 64
2.2.3 Handlungssicherheit 65

3. Entwurf der COJAC-Raumkomponente 68

3.1 Eine virtuelle Arbeitsumgebung für die kooperative Arbeit 69
3.1.1 Fokussieren eines virtuellen Platzes 71
3.1.2 Die Rollen von individuellem Arbeitsplatz und Privatgebrauch 72
3.1.3 Die räumliche Beziehung von Einzelarbeitsplatz und gemeinsamer Umgebung 74

3.2 Das fachliche Modell 76
3.2.1 Personen 77
3.2.2 Ressourcen 78
3.2.3 Einzelarbeitsplätze und Räume 80
3.2.3.1 Einzelarbeitsplatz 81
3.2.3.2 Raum 82
3.2.3.3 Raumbeschreibung 84
3.2.3.4 Inhalts- und Positionenverzeichnis 85
3.2.3.5 Inventarliste 85
3.2.3.6 Gebäude 86
3.2.3.7 Gebäudeplan 88

3.3 Das Benutzungsmodell 89
3.3.1 Visualisierung der Plätze 90
3.3.2 Fokussieren eines Raumes 94
3.3.3 Ansichten eines Raumes – Scrollen und "Zoomen" 95
3.3.4 Reservieren einer Ressource durch Markieren 97
3.3.5 Awareness 98
3.3.6 Bewegen von Dingen 101
3.3.7 Inventar 103

3.4 Die erweiterte COJAC Raumkomponente 105
3.4.1 Gruppenwerkzeuge 105
3.4.2 Eingebettete Plätze in Räumen: Möbel und Stapel 107
3.4.2.1 Die Platz-Hierarchie der gemeinsamen Umgebung 108
3.4.2.2 Die Handhabung von eingebetteten Plätzen im RoomExplorer 110
3.4.3 Abschließbare Räume 110
3.4.4 Gruppieren von Räumen: Etagen und Trakte 112
3.4.5 Explizite synchrone Kommunikation in Räumen: Chat- / Audio-Verbindung 113
3.4.6 Gebäudeplan-Menü 114
3.4.7 Web-Interface 116

4. Konstruktion der COJAC Basis-Raumkomponente 119

4.1 Das technische Modell der gemeinsamen Umgebung 121
4.1.1 Room (Raum) 122
4.1.2 Building (Gebäude) 126
4.2 Gruppenwerkzeug RoomExplorer 129
4.2.1 RoomExplorerServer 132
4.2.2 Ein Schnappschuss des Gebäudezustands: RoomExplorerSnapshot 135
4.2.3 RoomExplorerClient 139
4.2.4 Die lokale Werkzeugkomponente: RoomExplorer und RoomTop 141

5. Zusammenfassung und Ausblick 145

5.1 Die Idee hinter der Raumkomponente 145
5.2 Die Handhabung und Präsentation der Raumkomponente 146
5.3 Die Konstruktion der Raumkomponente 147
5.4 Ausblick 147

Literaturverzeichnis

Verzeichnisse

Anhang

0. Einleitung

Der WAM-Ansatz[2] mit seinem Leitbild des *Arbeitsplatzes für eigenverantwortliche Expertentätigkeit* ist ursprünglich auf die Entwicklung von Einzelarbeitsplätzen fokussiert. Ein wesentlicher Bestandteil qualifizierter Arbeit ist die Kooperation, die eine Erweiterung der Einzelarbeitsplätze um Kooperationsunterstützung notwendig macht. Die Integration der i.d.R. sehr spezialisierten Einzelarbeitsplätze zu einem kooperationsunterstützenden Gesamtsystem stellt sich als neues Problem dar – sowohl unter dem Aspekt des Benutzungsmodells als auch der technischen Umsetzung.

Im Rahmen der Diplomarbeiten [Freund00] und [Breitling00] sowie dieser Arbeit werden Konzepte zur Kooperationsunterstützung ausgearbeitet. Die Autoren der drei Diplomarbeiten und die gemeinsame Erstbetreuerin Dr. Ingrid Wetzel bilden das COJAC-Projekt, dessen Ziel ist, Komponenten zur Kooperationsunterstützung für das JWAM-Framework zu entwickeln.
Die Schwerpunkte der drei Diplomarbeiten liegen auf:

- expliziter Kooperationsunterstützung über ein Postversandsystem,
- Unterstützung von kooperativen Arbeitsprozessen über Vorgangsmappen und Vorgangsmonitor,
- situative Koperationsunterstützung über gemeinsame virtuelle Räume.

Diese Arbeit befasst sich mit dem dritten Punkt.

Voraussetzung ist, die kooperative Arbeit in Räumen besser zu verstehen, um sie durch virtuelle Räume angemessen zu unterstützen. Grundlage bietet hierfür eine Vorgehensweise nach dem WAM-Ansatz (siehe z.B. [WAM98], [Gryczan96], [KrabbelWetzel00]). Der WAM-Ansatz beinhaltet eine unterstützende Sichtweise auf die Softwareentwicklung.

Räume haben in der physischen Welt eine große Bedeutung für die Organisation der kooperativen Arbeit. Ein Großteil der Kooperation findet in den Räumen einer gemeinsamen Umgebung statt. Die Räume sind der Rahmen, in den Ressourcen und Handlungen eingebettet sind.

Dabei ermöglichen Räume sehr unterschiedliche Kooperations- und Arbeitsstile: individuell oder kooperativ, synchron oder asynchron, formal oder informal, explizit oder implizit usw. Diese Freiheit ermöglicht den Personen, sich in ihrem Handeln nach eigenem Ermessen auf konkrete Arbeitssituationen einzustellen.

Aufgrund ähnlicher Erfahrungen mit den physischen Bedingungen der Räume besitzen die kooperierenden Personen ein gemeinsames Verständnis ihrer gemeinsamen Umgebung sowie der Handlungsmöglichkeiten darin. Das erleichtert die Kommunikation über die gemeinsame Umgebung. Da zudem die Handlungen der anderen Personen beobachtet und die eigenen daran ausgerichtet werden können, ermöglichen Räume den Personen, sich situativ zu koordinieren und Konventionen bei der Kooperation zu entwickeln.

[2] WAM steht für die Entwurfsmetaphern Werkzeug, Automat und Material.

Die an Räume geknüpften Konventionen sind ein wichtiges Mittel, um verschiedene Sichtweisen und Arbeitsstile miteinander zu verbinden.

Angesichts der Bedeutung physischer Räume für die Kooperation scheint es naheliegend, in einem kooperationsunterstützenden Softwaresystem virtuelle Räume einzuführen.

Vielen Softwaresystemen liegt bereits die Vorstellung einer räumlichen Umgebung zugrunde. So ist z.B. die Umgebung des virtuellen Einzelarbeitsplatzes der Ort, an dem sich die Arbeitsmittel und -gegenstände für die individuelle Arbeit befinden.
In kooperationsunterstützenden Systemen existieren zudem Umgebungen, die von mehreren Personen benutzt werden. Der virtuelle Raum wird dadurch über die Grenzen des Einzelarbeitsplatzes hinaus erweitert. Hier bietet die Modellierung von virtuellen Räumen die Möglichkeit, die gemeinsame Umgebung in Analogie zur physischen Welt zu gestalten.

Dabei stellt sich die Frage, welche Eigenschaften der physischen Räume für die Kooperationsunterstützung relevant sind und wie sich diese auf ein Modell der virtuellen Räume übertragen lassen. Hier finden sich Ansätze in der CSCW-Literatur (z.B. bei [Mansfield98], [Kuhn96] und [HarrisonDourish96]) in denen z.B. die gegenseitige Wahrnehmung, die Aufteilung der Umgebung in getrennte Bereiche, das Konzept der Nähe usw. genannt werden.

Neben den räumlichen Merkmalen, die die virtuellen Räume aufweisen sollen, sind es vor allem die von den Personen mit den Räumen verknüpften Konventionen, die die kooperative Arbeit unterstützen. Diese fachliche Seite der Räume ist ebenfalls Bestandteil eines Modells der virtuellen Räume. Dabei darf die explizite Modellierung die Gestaltung der Arbeit nach eigenem Ermessen nicht einschränken, sondern soll diese vielmehr unterstützen. Auch hierzu gibt es Hinweise aus der CSCW-Literatur sowie einige konkrete Modelle und Lösungen (z.B. bei [HarrisonDourish96], [RoockWolf98], [GreenbergRoseman98] und [Mark97]) .

Eine weitere Anforderung an ein Modell der virtuellen Räume ist die Kompatibilität zu etablierten Konzepten zur Unterstützung individueller und kooperativer Arbeit. Hierzu zählen z.B. das Konzept des Einzelarbeitsplatzes und Konzepte zur Untertützung spezieller Kooperationsformen, z.B. der expliziten asynchronen Koopreation via Postversandsystem.

Ziel dieser Arbeit ist der Entwurf eines solchen Modells virtueller Räume und die konkrete Umsetzung des Entwurfs in einer Raumkomponente für das JWAM-Framework.

Beim Entwurf des Benutzungsmodells der Raumkomponente ist vor allem die Frage der Visualisierung interessant. Hier bietet die Desktopmetapher eine Anregung zur Gestaltung, deren Bedeutung für den WAM-Ansatz in [Lippert99] herausgearbeitet wird.

Frameworks sind wichtiger Bestandteil der Entwicklung großer Softwaresysteme. Sie geben eine allgemeine Architektur vor, auf deren Basis spezielle Anwendungssysteme entwickelt werden können. Frameworks ermöglichen die Wiederverwendung eines Entwurfs und geben

konkrete Entwurfentscheidungen vor. Mithilfe von Frameworks können Softwaresysteme aus vergleichbaren Anwendungsbereichen ähnlich strukturiert und schneller entwickelt werden (siehe z.B. [Gamma97], [Szyperski99] und [Schmid97]).

Das JWAM-Framework ist ein in Schichten aufgebautes, komponentenbasiertes Framework in der Programmiersprache Java (siehe z.B. [JWAM99 1], [Bäumer98]). Die dort realisierten Konzepte entstammen dem WAM-Ansatz.

Bei der Realisierung der Raumkomponente ist dem im JWAM-Framework verfolgten Komponentengedanken zu folgen. Die Raumkomponente soll über Spezialisierung und Komposition an die Anforderungen des jeweiligen Anwendungsbereichs angepasst werden können. Dabei sind die Stellen (HotSpots) zu spezifizieren, an denen die Anpassung vorgenommen werden kann.

0.1 Ziele der Arbeit

Ziel dieser Arbeit ist, auf der Basis des JWAM-Frameworks eine Raumkomponente zur Kooperationsunterstützung zu entwerfen und zu implementieren.

Die konzeptionellen Grundlagen existieren bereits in den Konzepten der Arbeitsumgebung und des Arbeitsplatzes, die hier dem WAM-Ansatz entstammen. Deren Rolle in einem kooperationsunterstützenden Kontext muss jedoch differenzierter ausgearbeitet werden. Individuelle und kooperative Arbeit sollen dabei gleichermaßen unterstützt werden.

Um die Bedeutung von Räumen – insbesondere für die informale Kooperation – deutlich zu machen und um Anhaltspunkte zu geben, an denen sich die Beschreibung der weiteren Modellierung orientieren kann, sollen beispielhafte Kooperationssituationen gefunden und beschrieben werden.

Die konkrete fachliche Modellierung der gemeinsamen Umgebung bildet die Grundlage für den weiteren Entwurf der Raumkomponente und ist damit ein besonderes Ziel. Wichtigstes strukturierendes Element der gemeinsamen Umgebung sind die virtuellen Räume. Darüberhinaus sind weitere fachliche Konzepte zu identifizieren, die der Kooperationsunterstützung durch eine Raumkomponente dienen.

Der Entwurf des Benutzungsmodells der Raumkomponente stellt einen großen Themenschwerpunkt dieser Arbeit dar.
Die Visualisierung der virtuellen Räumlichkeiten ist dabei als ein wichtiges Problem zu nennen. Diese sollte kein "überdurchschnittliches" räumliches Vorstellungsvermögen beim Benutzer voraussetzen – auf der anderen Seite sollte die Visualisierung sich möglichst dezent in das bestehende, übliche Look-And-Feel einpassen und z.B. nicht im Widerspruch zum bereits existierenden JWAM-Desktop stehen.
Ein Blick auf bestehende CSCW-Applikationen, denen ein räumliches Modell zugrundeliegt, soll Anregungen für den Entwurf des Benutzungsmodells liefern.

Über den Entwurf der zu realisierenden Raumkomponente hinaus sind Visionen für mögliche Erweiterungen derselben zu entwickeln.

Für die Realisierung der Raumkomponente auf der Basis des JWAM-Frameworks müssen schließlich verschiedene softwaretechnische Probleme gelöst werden. So muß z.B. eine geeignete verteilte Architektur für die Raumkomponente gefunden werden. Insbesondere muss eine Lösung gefunden werden, wie mit einem lokalen Werkzeug auf die entfernten Räumlichkeiten zugegriffen werden kann.
Daneben gilt es, sich in das JWAM-Framework einzuarbeiten, bestehende JWAM-Komponenten zu nutzen und ggf. anzupassen.

0.2 Aufbau der Arbeit

Im ersten Kapitel möchte ich den **Kontext der Arbeit** vorstellen, in dem ich sie eingebettet sehe.
Der **WAM-Ansatz** prägt diese Arbeit ganz entscheidend, daher stelle ich diesen an den Anfang. In einer knappen Einführung stelle ich die wichtigsten Begriffe und Konzepte des WAM-Ansatzes vor, um sie in der weiteren Arbeit verwenden zu können.
Das Thema **Kooperationsunterstützung** / **CSCW** behandle ich im nächsten Unterkapitel. Dort erläutere ich weitere Begriffe und gebe einen kurzen Überblick über die Historie des CSCW-Forschungszweigs und die Klassifikation von CSCW-Applikationen.
Im folgenden Kapitel stelle ich drei **konkrete CSCW-Applikationen** vor, in denen Kooperation durch eine **gemeinsame räumliche Umgebung** unterstützt wird. Auf der Basis der Konzepte, die diesen CSCW-Applikationen zugrundeliegen und weiteren Ansätzen aus der CSCW-Diskussion erarbeite ich schließlich das Verständnis der Begriffe "Ort" und "Platz", auf das ich mich in dieser Arbeit berufe.
Anschließend beschreibe ich das **JWAM-Framework.** Im JWAM-Framework werden die Konzepte des WAM-Ansatzes in konkreten Komponenten verwirklicht. Es bildet die technische Grundlage für den konstruktiven Teil dieser Arbeit.
Am Ende dieses Kapitels stelle ich die **COJAC- und JWAM-Komponenten** zur Kooperationsunterstützung vor, von denen die hier entwickelte COJAC-Raumkomponente eine ist.

Im zweiten Kapitel betrachte ich die **fachliche Motivation für eine Raumkomponente** genauer.
Mit einem **Beispiel aus dem Buchverlag** stelle ich typische Kooperationssituationen vor, an denen wichtige Konzepte erkennbar sind, die eine Raumkomponente für die Unterstützung solcher Kooperationssituationen umsetzen sollte.

Das dritte Kapitel dieser Arbeit befasst sich mit dem **Entwurf der COJAC-Raumkomponente**.
Dazu betrachte ich zunächst das Konzept einer **virtuellen Arbeitsumgebung für die kooperative Arbeit** und entwickle ein differenzierteres Verständnis ihrer Struktur.

Im Anschluss daran beschreibe ich den Entwurf der **COJAC Basis-Raumkomponente**. Der erste Teil dieses Entwurfs befasst sich mit dem **fachlichen Modell** der Raumkomponente, in dem die Konzepte aus dem zweiten Kapitel konkretisiert werden. Der zweite Teil des Entwurfs betrifft das **Benutzungsmodell**, in dem ich eine detaillierte Vision der Handhabung und Präsentation der Raumkomponente entwickele.
Am Ende dieses Kapitels gebe ich in Form von recht detaillierten Visionen einen Ausblick auf eine **erweiterte Raumkomponente**. Eine konzeptionelle Erweiterung sind dabei Gruppenwerkzeuge, mit denen auf Materialien in der gemeinsamen Umgebung gearbeitet werden kann – quasi WAM-Groupware.

Auf der Basis des im vorhergehenden Kapitel beschriebenen Entwurfs erläutere ich im vierten Kapitel die **Konstruktion der COJAC Basis-Raumkomponente**.
Dazu beschreibe ich zunächst das **technische Modell**, das in engem Zusammenhang zum fachlichen Modell steht.
Anschließend folgt die Beschreibung der Konstruktion des **Gruppenwerkzeugs RoomExplorer**, mit dem auf die Räume der gemeinsamen Umgebung zugegriffen werden kann.
Den Abschluss der Arbeit bildet eine **Zusammenfassung** der Ergebnisse der Arbeit und ein **Ausblick** auf offene Fragen.

0.3 Konventionen

Der neuen Rechtschreibreform zolle ich über die Schreibweise "ss" statt "ß" nach einem kurzen Vokal meinen Tribut. Weitere Neuerungen in der Rechtschreibung habe ich nicht übernommen.

Wichtige Begriffe setze ich bei der ersten Erwähnung kursiv. Spreche ich auf einer Meta-Ebene über Begriffe, stehen diese in Anführungszeichen.

Die Fußnoten sind kapitelweise durchnumeriert, um dreistellige Zahlen weitgehend zu vermeiden.
Aufgrund eines Fehlers oder Features von Word 97 erscheinen einige Fußnotentexte erst auf der Folgeseite hinter der Fußnote. Das ist irritierend aber leider nicht zu umgehen – ich bitte um Verständnis.

0.4 Danksagung

An dieser Stelle möchte ich mich bei all den Menschen bedanken, die mir bei der Erstellung dieser Arbeit auf verschiedene Weise behilflich waren.
Dr. Ingrid Wetzel danke ich für die Erstbetreuung sowie für die zahlreichen und intensiven Diskussionen. Prof. Dr. Arno Rolf danke ich für die Übernahme der Zweitbetreuung.
Insbesondere gilt mein Dank den Mitstreitern des COJAC-Teams, Holger Breitling und Mirko Freund, sowie Henning Wolf, der sich an den Diskussionen des COJAC-Teams über einen längeren Zeitraum beteiligt hat.

Prof. Dr. Heinz Züllighoven verdanke ich einige wertvolle Anregungen. Martin Lippert und Mareile Wilkens möchte für die hilfreichen Hinweise aus dem Kontext ihrer Arbeiten Dank sagen.
Bei Jeannette Popken bedanke ich mich sehr für die ausführlichen Auskünfte zum Buchverlag und die moralische Unterstützung über die lange Zeit. Zudem danke ich ihr und meinem Vater für das Korrekturlesen der Arbeit.

1. Einbettung der Arbeit

Im folgenden stelle ich zunächst den Methodenrahmen vor und die damit verknüpfte Sicht auf die Softwareentwicklung, die dieser Arbeit zugrundeliegt und in allen Teilen prägt.
Anschließend gebe ich eine knappe allgemeine Einführung in die Problematik der Kooperationsunterstützung und den Forschungsbereich CSCW[3]. Dem folgt eine detailliertere Betrachtung räumlicher Konzepte in der CSCW-Diskussion und ein Vergleich dreier Groupware-Systeme[4], die derartige Konzepte umsetzen[5].
Die Problematik der Frameworkentwicklung und das JWAM-Framework sind der technologische Kontext, in den der konstruktive Teil dieser Arbeit eingebettet ist[6].
Den Abschluss dieses Kapitels bilden der unmittelbare Entstehungskontext dieser Arbeit, das COJAC-Projekt, sowie weitere Arbeiten des engeren WAM- und JWAM-Umfeldes[7].

1.1 Der WAM-Ansatz

Der WAM-Ansatz wird an der Universität Hamburg am Arbeitsbereich Softwaretechnik des Fachbereichs Informatik seit Beginn der Neunziger Jahre vertreten und weiter ausgearbeitet.
Das zentrale Handbuch zum WAM-Ansatz ist [WAM98]. Weiterhin sind noch [Kilberth94] als erstes umfassenderes Werk, sowie [Gryczan95] und [Bäumer98] besonders zu erwähnen. Darüberhinaus ist im engeren Umfeld des WAM-Ansatzes eine Vielzahl an weiteren Publikationen entstanden, darunter zahlreiche Studien- und Diplomarbeiten[8].

Der WAM-Ansatz ist ein *Methodenrahmen* für objektorientierte, evolutionäre Softwareentwicklung. Als Methodenrahmen soll er Softwareentwicklern helfen, ihre eigene Methode bei der Konstruktion eines Softwaresystems für einen bestimmten Anwendungsbereich zu entwickeln.
Er umfasst:

"
- eine Sicht der objektorientierten Anwendungsentwicklung,
- eine Sammlung bewährter Konstruktions-, Analyse- und Dokumentationstechniken,
- eine Beschreibung zusammenpassender Konzepte,
- eine Auswertung unterschiedlicher und umfangreicher Projekterfahrungen,
- eine Anleitung, um eine konkrete Konstruktionstechnik und eine dazu passende Vorgehensweise zu entwickeln."[9]

Der WAM-Ansatz soll im folgenden genauer beschrieben werden.

---

[3] Kap.1.2, S.23 – CSCW steht für Computer Supported Cooperative Work
[4] Kap.1.2, S.25 – Groupware bezeichnet Softwaresysteme zur Unterstützung von Gruppenarbeit
[5] Kap.1.3, S.27
[6] Kap.1.4, S.43
[7] Kap.1.5, S.54
[8] viele der Publikationen sind downloadbar unter: swt-www.informatik.uni-hamburg.de/publications/
[9] [WAM98], S.19f.

1.1.1 Softwareentwicklung als dokumentenbasierter Kommunikations- und Lernprozess

Als grundlegendes Merkmal des Softwareentwicklungsprozesses fordert der WAM-Ansatz *Anwendungsorientierung*. Das bedeutet im einzelnen, dass die Funktionalität des Softwaresystems sich an den Aufgaben im Anwendungsbereich zu orientieren hat, die Handhabung des Systems benutzergerecht ist und die im System festgelegten Abläufe und Schritte sich an die Erfordernisse der Arbeitssituation anpassen lassen[10].

Für die Entwickler bedeutet das, die Aufgaben des Anwendungsbereichs zu identifizieren und zu verstehen. Softwareentwicklung ist ein *Kommunikations- und Lernprozess* zwischen Entwicklern und Benutzern[11].

Dieser gegenseitige Lernprozess wird durch eine *evolutionäre Systementwicklung* ermöglicht, in der ein Softwaresystem in mehreren Entwicklungszyklen an sich verändernde Anforderungen und Rahmenbedingungen angepasst wird[12].

Entwickler und Benutzer des Systems stehen im ständigen Dialog. Dabei erarbeiten und verwenden sie gemeinsam *Dokumente*, die den Anwendungsbereich und das zu entwickelnde System allgemeinverständlich modellieren.
Primäres Ziel der Arbeit mit den Dokumenten ist, den Kommunikations- und Lernprozess zu fördern.
Die Dokumente durchlaufen zahlreiche *Autor-Kritiker-Zyklen*, wodurch bereits bei der Ausarbeitung der Dokumente *konstruktive Qualitätssicherung* betrieben wird[13].

Szenarios spielen im WAM-Ansatz eine zentrale Rolle[14]. Sie gehören mit den *Glossaren* und *Systemvisionen* zu den "klassischen" Dokumenttypen des dokumentbasierten Entwicklungsprozesses nach WAM.

Begriff 1: Szenario

Ein Szenario ist ein Prosatext in der Sprache der Anwendung, der eine Ist-Situation beschreibt. Gegenstand der Beschreibung ist, wie Aufgaben im Anwendungsbereich erledigt werden.

Szenarios dienen der Aufgabenanalyse. Die technische Realisierung bzw. Realisierbarkeit in einem System spielt in ihnen keine Rolle.
Man unterscheidet – bei zunehmendem Detaillierungsgrad – zwischen Überblicksszenario, Aufgabenszenario und Handlungsstudie[15].

[10] [WAM98], S.260f.
[11] [WAM98], S.265f.
[12] [WAM98], S.273
[13] [WAM98], S.276ff.
[14] [WAM98], S.1216
[15] [WAM98], S.1217f.

Die in den Szenarios verwendeten und alle weiteren relevanten Begriffe des Anwendungsbereichs werden in einem Glossar verzeichnet. Das Glossar ist gewissermaßen ein Wörterbuch der jeweiligen Fachsprache, die einem Anwendungssystem zugrunde liegt[16].

Bei der Dokumentation des Begriffsmodells haben sich zudem *CRC-Karten* bewährt. CRC steht für Class-Responsibility-Collaboration. Auf einer CRC-Karte werden die Dienstleistungsbeziehungen für eine bestimmte Klasse vermerkt. Das sind die Dienstleistungen, die sie erbringt, und die Klassen, von denen sie Dienstleistungen in Anspruch nimmt[17].

Mithilfe von Systemvisionen soll die kreative Umsetzung von der analysierten Ist-Situation zum angestrebten Anwendungssystem stattfinden. Systemvisionen sind vor allem für die Entwickler interessant, um eine gemeinsame Vision des Anwendungssystems in ihren Köpfen entstehen zu lassen. Anwendern fehlt zumeist das softwaretechnische Vorstellungsvermögen[18].
Eine Systemvision beschreibt als kurzer Prosatext, wie fachliche Aufgaben mithilfe des zu entwickelnden Anwendungssystems unter Verwendung von Werkzeugen, Automaten und Materialien erledigt werden können[19].

Bei der Analyse von kooperativer Arbeit werden neben den genannten Dokumenttypen *Kooperationsbilder* verwendet. Sie dienen dazu, komplexe Zusammenhänge zu visualisieren und allen Beteiligten einen Überblick über die in den Szenarios geschilderten Aufgaben zu verschaffen.

Begriff 2: Kooperationsbild

> "Ein Kooperationsbild [...] ist eine visuelle Darstellung einer kooperativen Arbeitssituation, besonders bei übergreifenden Aufgaben [...]. Dabei werden allgemeinverständliche Piktogramme verwendet. Kooperationsbilder stellen dar, wie Personen arbeitsteilig zusammenarbeiten."[20]

Krabbel und Wetzel beschreiben eine besondere Form des Kooperationsbildes, das *task picture*[21]. Diese dienen dazu, Unterschiede zwischen ähnlichen Arbeitsplätzen zu diskutieren. Sie beschreiben die involvierten Personen, die verwendeten Materialien und die jeweilige Art der Interaktionen. Task pictures werden zu einzelnen Szenarios erstellt. Der Fokus liegt bei ihnen auf Aufgaben, die innerhalb einer kleinen Einheit bearbeitet werden.

Beispiele für Szenarios und task pictures finden sich im zweiten Kapitel dieser Arbeit[22].

---

[16] [WAM98], S.1237
[17] [WAM98], S.314
[18] [WAM98], S.1255
[19] [WAM98], S.1247
[20] [WAM98], S.1301
[21] [KrabbelWetzel00], S.14
[22] Kap.2.1.1, S.59 und Kap.2.1.2, S.60

1.1.2 Softwareentwicklung als Modellierungsprozess

Im Projektverlauf werden verschiedene fachliche und softwaretechnische Modelle erstellt. Softwareentwicklung ist vor allem ein *Modellierungsprozess*. Die zentrale Idee des WAM-Ansatzes ist, ein *fachliches Modell* des Anwendungsbereichs auf der Grundlage der relevanten Gegenstände und Konzepte zu erstellen[23].

Die identifizierten Gegenstände und Konzepte werden über die charakteristischen Handlungen an und mit ihnen – den sogenannten *Umgangsformen* – beschrieben. Durch Abstraktion der konkreten Gegenstände und Umgangsformen wird ein Begriffsmodell in der Fachsprache des Anwendungsbereichs erstellt. Durch Generalisierung und Komposition werden die Begriffe zueinander in Beziehung gesetzt[24].
Das fachliche Modell ist Ausdruck dessen, wie der Anwendungsbereich von den beteiligten Benutzern und Entwicklern verstanden wird. Grundlage dieses Verständnisses bildet ein gemeinsames Leitbild mit seinen Entwurfsmetaphern (s.u.)[25].

Dabei wird *Strukturähnlichkeit* zwischen dem anwendungsfachlichen Begriffsgebäude und der Softwarearchitektur gefordert: Die softwaretechnischen Komponenten eines Softwaresystems entsprechen den Elementen des Modells des Anwendungsbereichs.
Für die Benutzer bringt die Strukturähnlichkeit den entscheidenden Vorteil, dass sie die Gegenstände ihrer Arbeit und die Begriffe ihrer Fachsprache im Anwendungssystem wiederfinden. Für die Entwickler liegt der Vorteil darin, dass sie die Abhängigkeiten zwischen fachlichen und softwaretechnischen Konzepten besser erkennen und leichter verstehen können[26].

Den Elementen des *Objekt-Metamodells*, das die für die objektorientierte Modellierung zur Verfügung stehenden Elemente und deren Verknüpfungen beschreibt, wird neben der technischen Interpretation auch eine fachliche Interpretation gegeben. Damit wird der Zusammenhang zwischen fachlichem Begriffsmodell und softwaretechnischem Modell (in der weiteren Arbeit auch kurz *technisches Modell* genannt) festgelegt:

[23] [WAM98], S.297f.
[24] [WAM98], S.312f.
[25] nach [Bäumer98], S.49
[26] [WAM98], S.263

fachliches Modell	softwaretechnisches Modell
Gegenstand	Objekt
Umgangsform	Operation
Begriff	Klasse
Generalisierung, Spezialisierung	Vererbung
Komposition	Aggregation, Assoziation
Begriffshierarchie	Klassenhierarchie

Tabelle 1: Zusammenhänge zwischen fachlichem und technischem Modell[27]

Die Elemente des fachlichen und des softwaretechnischen Modells lassen sich so weitgehend ohne Modellbruch aufeinander abbilden[28].

Ein rein fachliches Modell des Anwendungsbereichs gibt noch keine Antwort, wie ein interaktives Anwendungssystem gestaltet werden soll.
Neben dem Look&Feel ist vor allem das *Benutzungsmodell* von Bedeutung, also die Vorstellung, die sich Benutzer und Entwickler machen, auf welche Weise die Software in konkreten Arbeitssituationen verwendet werden kann[29].

Begriff 3: Benutzungsmodell

"Ein Benutzungsmodell ist ein fachlich orientiertes Modell darüber, wie Anwendungssoftware bei der Erledigung der anstehenden Aufgaben im jeweiligen Einsatzkontext benutzt werden kann.

Das Benutzungsmodell umfaßt eine Vorstellung von der Handhabung und Präsentation der Software aber auch von den fachlichen Gegenständen, Konzepten und Abläufen, die von der Software unterstützt werden.

Es ist sinnvoll, ein Benutzungsmodell auf der Grundlage eines Leitbilds mit Entwurfsmetaphern zu realisieren."[30]

Es verwundert nicht, dass das fachliche Modell ein wesentlicher Teil des Benutzungsmodells ist. Das fachliche Modell beschreibt die Arbeitsgegenstände. Das Benutzungsmodell beschreibt, wie diese Arbeitsgegenstände eingesetzt und mit welchen Mitteln sie bearbeitet werden können.

Dem Benutzungsmodell kommt eine entscheidende Bedeutung beim Entwurf von Anwendungssoftware zu. Der Benutzer kann die Software nur so effektiv einsetzen, wie es ihm sein Verständnis des Benutzungsmodells gestattet.

[27] [WAM98], S.46. und [Bäumer98], S.10
[28] [WAM98], S.45f.
[29] [WAM98], S.147
[30] [WAM98], S.148

Das Benutzungsmodell sollte in sich stimmig sein und eine Einheit von Form und Inhalt aufweisen. In diesem Sinne ist es sinnvoll, ein Benutzungsmodell auf der Grundlage eines Leitbilds mit Entwurfsmetaphern (s.u.) zu entwickeln[31].

Das Benutzungsmodell befindet sich zwischen Anwendungsbereich und verwendeter Technik in einem Spannungsfeld der Anforderungen (siehe Abb.1). In Bezug auf den Anwendungsbereich wird im WAM-Ansatz Anwendungsorientierung für ein Softwaresystem und somit für dessen Benutzungsmodell gefordert. Ein technikinduziertes Benutzungsmodell nach dem Motto "Hier haben wir die Technologie – benutzen wir sie!" wird dieser Anforderung sicher nicht gerecht.
Andererseits darf die verwendete Technik im Benutzungsmodell nicht übersehen werden. Zum einen hat sie – quasi als begrenzender Rahmen – maßgeblichen Einfluss auf das, was im Benutzungsmodell an sinnvollen Konzepten überhaupt vorstellbar ist. Es wäre z.B. nicht sinnvoll, ein virtuelles Postversandsystem vorzusehen, wenn keine technische Möglichkeit der Verteilung existieren[32].
Zum anderen sollten technische Gegebenheiten, soweit sie sich auf die Handhabung auswirken können, für den Benutzer transparent, d.h. durchschaubar sein. So wäre es z.B. nicht sinnvoll, wenn das Benutzungsmodell dem Benutzer suggeriert, ein technisch entferntes Material sei lokal vorhanden und – ungeachtet der Gefahr von Verbindungsproblemen – jederzeit verfügbar. Der Benutzer könnte sich im Fehlerfall kein Bild von den Ursachen machen und angemessen reagieren[33].

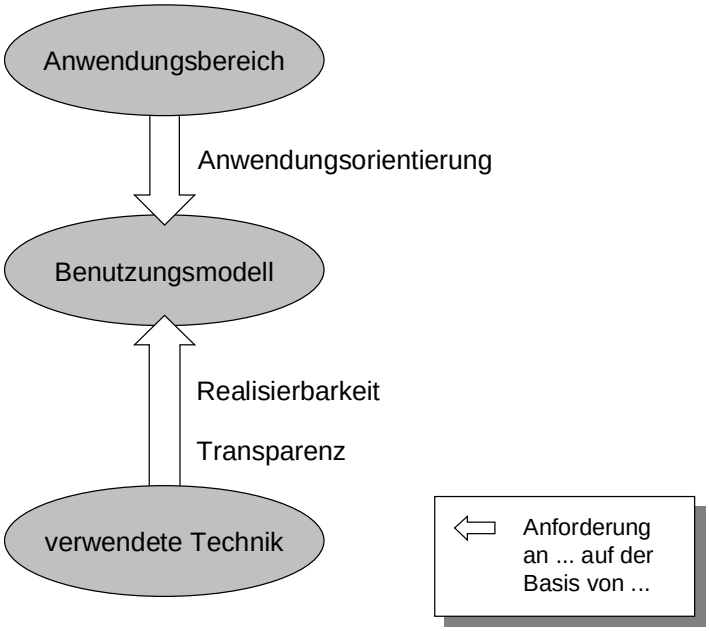


Abbildung 1: Das Benutzungsmodell im Spannungsfeld der Anforderungen

[31] [WAM98], S.147ff.
[32] [JWAM99 1], S.91
[33] [WAM98], S.881f. – die Frage der Transparenz in einer kooperativen, technisch verteilten Umgebung wird im Kapitel 3.1.3 auf S.74 ausführlicher diskutiert

Derzeit fehlen noch Gestaltungsrichtlinien, wie gute interaktive Software für die Unterstützung von Arbeit – insbesondere kooperativer Arbeit – aussehen sollte. Die konventionelle sequentielle Systemgestaltung wird eher einer ablauforientierten Sichtweise gerecht. Grafische Oberflächen bieten z.B. mit der Umsetzung der Desktopmetapher[34] Ansätze, wie die Einheit von Form und Inhalt realisiert werden könnte[35]. Weitere Ansätze finden sich z.B. in der CSCW-Forschung[36].

1.1.3 Das WAM-Leitbild mit seinen Entwurfsmetaphern

Die zwei historischen und konzeptionellen Wurzeln des WAM-Ansatzes sind zum einen die Arbeit von Softwareentwicklern, zum anderen die Büroarbeit im Finanz- und Dienstleistungsbereich. In diesem Kontext – der Büroarbeit qualifizierter Mitarbeiter – hat sich das explizite *Leitbild* des *"Arbeitsplatzes für eigenverantwortliche Expertentätigkeit"* bewährt[37]. Dies ist das häufigste Leitbild des WAM-Ansatzes und auch dieser Arbeit liegt es zugrunde.

Dem Leitbild liegt eine besondere Wertvorstellung von qualifizierter menschlicher Arbeit zugrunde, die sich in einer *unterstützenden Sichtweise* auswirkt:
Qualifizierte menschliche Arbeit orientiert sich zielgerichtet an vorhandenen Arbeitsmitteln und -gegenständen und nicht an vorgegebenen Routinen und Abläufen. Die Arbeitsschritte werden durch die vorhandenen Arbeitsmittel angemessen unterstützt.
Ein Experte wählt bei seiner Arbeit die Mittel situativ aus, um die sehr komplexen und häufig wechselnden Aufgaben zu erledigen. Obwohl er aufgrund seiner Qualifikation und Erfahrung planvoll und zielgerichtet vorgeht, können die Arbeitsabläufe nicht formalisiert werden – die Kontrolle über die Handlungen muss also beim Menschen verbleiben[38].
Kurz gesagt: Menschliche Arbeit wird als weitgehend unersetzlich betrachtet und soll durch die Anwendungssoftware unterstützt werden[39]. Der WAM-Ansatz kann mit seiner unterstützenden Sichtweise daher als Gegenpol zur ablaufsteuernden Sichtweise gesehen werden[40].

Das Leitbild des WAM-Ansatzes wird durch *Entwurfsmetaphern* konkretisiert. Als bildhafte, gegenständliche Vorstellungen machen sie den Anwendungsbereich durchschaubarer und die Handhabung und Funktionalität des entwickelten Systems für alle Beteiligten verständlicher. Die Entwurfsmetaphern des WAM-Ansatzes haben immer auch eine technische Interpretation in Form von *Konstruktionsanleitungen* und *Entwurfsmustern*, auf deren Basis sie sich effizient implementieren lassen[41].

---

[34] [Lippert99]
[35] [WAM98], S.149f.
[36] Kap.1.3, S.27
[37] [WAM98], S.10 und S.167
[38] [WAM98], S.167ff.
[39] [WAM98], S.167
[40] [Rolf98], S.97
[41] [WAM98], S.165 und S.233f.

Von zentraler Bedeutung für den WAM-Ansatz sind die namensgebenden Entwurfsmetaphern Werkzeug, Automat und Material.

Begriff 4: Werkzeug

"Werkzeuge sind Gegenstände, mit denen Menschen im Rahmen einer Aufgabe Materialien verändern oder sondieren können.

Werkzeuge eignen sich meist für verschiedene fachliche Zwecke und für die Arbeit an unterschiedlichen Materialien. Sie müssen geeignet gehandhabt werden.

Werkzeuge vergegenständlichen wiederkehrende Arbeitshandlungen.

Viele konzeptionelle Eigenschaften von (Hand-) Werkzeugen lassen sich auf Softwarewerkzeuge übertragen. Eine direkte Abbildung der Handhabung und Gestalt ist aber selten sinnvoll."[42]

Werkzeuge präsentieren nicht nur das bearbeitete Material, sondern geben dem Benutzer Feedback über dessen Aktivitäten – in einem kooperativen Kontext u.U. auch über die Aktivitäten anderer Benutzer (dies wird oft als Awareness bezeichnet).

Werkzeuge sollten zweckdienlich gestaltet sein und dem Benutzer den Eindruck vermitteln, er würde direkt auf den Materialien arbeiten. Dies heißt nicht, dass sie konzeptionell verborgen werden sollten. Spätestens in einer Fehlersituation sollte dem Benutzer die Existenz des Werkzeugs bewusst werden können[43].

Beispiele für Werkzeuge sind Formular-Editoren, Auflister, Taschenrechner usw.

Für die Konstruktion von Werkzeugen existieren spezielle Entwurfsmuster – z.B. zur Trennung von Handhabung und Präsentation in Komponenten[44], die über das Beobachtermuster[45] lose gekoppelt werden. Viele Anteile der Werkzeug-Konstruktion sind generisch und daher in das JWAM-Framework eingegangen[46].

[42] [WAM98], S.177
[43] [Apcon T&M o.J]
[44] [WAM98], S.479ff.
Die genaue Beschreibung der Werkzeugkonstruktion ist sehr komplex und würde an dieser Stelle zu weit führen. Soweit notwendig, werde ich die Entwurfsmuster zur Werkzeugkonstruktion im vierten Kapitel erläutern. (Kap.4.2.4, S.141)
[45] [Gamma97], S.257
[46] [Apcon T&M o.J] und siehe Package de.jwam.handling.toolconstruction (die Verweise auf Packages beziehen sich auch in der weiteren Arbeit auf die Version JWAM 1.5.0 vom September 2000)

Begriff 5: Material

"Materialien sind Gegenstände, die im Rahmen einer Aufgabe Teil des Arbeitsergebnisses werden.

Materialien werden durch Werkzeuge und Automaten bearbeitet und verkörpern fachliche Konzepte. Sie müssen für die Bearbeitung geeignet sein.

Die Eigenschaften vorhandener Arbeitsgegenstände lassen sich oft sinnvoll auf Softwarematerialien übertragen."[47]

Auf Materialien kann niemals direkt, sondern immer nur über geeignete Werkzeuge zugegriffen werden.

Beispiele für Materialien sind Formulare, Notizen, Bankkonten usw.

Materialien und Werkzeuge werden über *Aspekte* lose gekoppelt. Aspekte definieren eine Verantwortlichkeit und können als "Dienstleistungsbündel" aufgefasst werden[48]. Im JWAM-Framework werden Aspekte über Schnittstellen[49] konstruiert. Materialien implementieren eine oder mehrere Aspekt-Schnittstellen und werden von geeigneten Werkzeugen ausschließlich über diese Aspekt-Schnittstellen verwendet.

Begriff 6: Automat

"Automaten sind im Rahmen einer zur erledigenden Aufgabe ein Arbeitsmittel, um Material zu bearbeiten. Sie erledigen lästige Routinetätigkeiten als eine definierte Folge von Arbeitsschritten mit festem Ergebnis ohne weitere äußere Eingriffe.

Automaten laufen unauffällig im Hintergrund, wenn sie einmal vom Benutzer oder von der Arbeitsumgebung gestartet sind.

Sie können auf ihren Zustand überprüft und im vorgegebenen Rahmen eingestellt werden."[50]

Automaten arbeiten wie Werkzeuge auf Materialien. Allerdings werden sie nicht wie diese interaktiv gehandhabt. Nachdem sie gestartet worden sind verrichten sie ihre Aufgabe ohne weiteres Zutun des Benutzers. Ggf. werden sie durch den Benutzer kontrolliert[51].
Die Operationen, die ein Benutzer mithilfe eines Werkzeugs ausführt, sind aus dessen Sicht atomar. Bei der Benutzung eines Automaten geht die Kontrolle über den festgelegten Handlungsablauf zeitweise an den Automaten über[52].

---

[47] [WAM98], S.180
[48] [WAM98], S.427
[49] Eine Schnittstelle ist einer vollständig abstrakten Klasse vergleichbar, die ausschließlich abstrakte Methoden definiert bzw. erbt – d.h. sie enthält ausschließlich die Signaturen dieser Methoden, nicht deren Implementation. Die Semantik der Methoden kann vertraglich spezifiziert sein.
In Java sind Schnittstellen eigenständige Modellelemente. Sie werden von Klassen implementiert. Dabei kann eine Klasse mehrere Schnittstellen implementieren ([Flanagan98], S.81f.).
[50] [WAM98], S.187
[51] [WAM98], S.186
[52] [WAM98], S.401f.

Einige Automaten können – bevor sie gestartet werden – über ein Einstellwerkzeug konfiguriert werden. Andere Automaten laufen "im Hintergrund", ohne dem Benutzer unmittelbar aufzufallen.

Diese "verborgenen" Automaten zeigen, wie in einem WAM-System selbst Komponenten, die nur der Entwickler "sieht", für diesen eine gegenständliche Interpretation haben können.
Automaten dienen nicht nur zur Automatisierung lästiger menschlicher Routinehandlungen, sondern auch zur Kapselung technischer Schnittstellen und zur Abbildung von technischen Prozessen[53].

Beispiele für Automaten sind Postein- und -ausgangsautomaten[54], "Schnüffler"[55] usw.

Die Kopplung von Automaten und Materialien geschieht konstruktiv – wie bei der Kopplung von Werkzeugen und Materialien – über Aspekte.

Mit den Entwurfsmetaphern Werkzeug, Automat und Material ist das Repertoire an Entwurfsmetaphern des WAM-Ansatzes noch nicht erschöpft.
In vielen Anwendungskontexten nehmen Behälter eine besondere Rolle unter den Materialien ein, die eine speziellere Entwurfsmetapher als schlicht Material rechtfertigen.

Begriff 7: Behälter

"Ein (fachlicher) Behälter kann Materialien aufnehmen, verwalten, ordnen und herausgeben. Dazu führt der Behälter oft Verzeichnisse.

In einem Behälter können viele gleichartige oder ein definierter Satz von unterschiedlichen Gegenständen verwahrt werden.

Behälter vergegenständlichen oft Prozesse als sogenannte Vorgänge (z.B. Kreditakte) und dienen der Kooperation und Koordination."[56]

Behälter besitzen meist charakteristische Umgangsformen, wie z.B. "Material einfügen", "Material entnehmen", "Anzahl der enthaltenen Materialien ermitteln", "ermitteln ob Behälter leer / voll" usw., und werden auf sehr ähnliche Weise modelliert. Diese generischen Anteile der Konstruktion von Behältern sind in das JWAM-Framework eingegangen[57].

Beispiele für Behälter sind Ordner, Mappen, Arbeitsflächen usw.

Die vorgestellten Entwurfsmetaphern beziehen sich ausschließlich auf die Arbeitsgegenstände und -mittel des Anwendungsbereichs, also auf Objekte.

---

[53] [WAM98], S.403
[54] [Freund00], S.66ff.
[55] kleine Automaten, die eine Registratur nach bestimmten Dokumenten durchsuchen – siehe z.B. TextSniffer in Package de.jwamx.handling.registry
[56] [WAM98], S.189
[57] siehe package de.jwam.handling.containerconstruction

Neben diesen Objekten werden aber auch anwendungsfachliche Werte – sogenannte *Fachwerte* – benötigt, um Dinge zu identifizieren, zu charakterisieren, abzuzählen und zu ordnen und als messbare Größen zu repräsentieren[58].
Beim Fachwert-Konzept handelt es sich nicht um eine Entwurfsmetapher. Dennoch sind Fachwerte wichtige Bestandteile des fachlichen Modells und verfügen – wie die Entwurfsmetaphern – auch über eine technische Interpretation in Form von Entwurfsmustern.

Werte sind im Gegensatz zu Objekten abstrakt und absolut. Sie können nicht unmittelbar, sondern nur über ihre Repräsentation erkannt werden. Von einem Objekt unterscheidet sich ein Wert vor allem in den folgenden Merkmalen:

Wert	Objekt
ist zeit- und ortlos	existiert in Raum und Zeit
abstrakt, ohne Identität, nur Gleichheit	konkret, mit Identität und Gleichheit
definiert oder undefiniert, aber nicht veränderlich	veränderbar bei Wahrung der Identität

Tabelle 2: Eigenschaften von Wert und Objekt[59]

In Bezug auf diese Merkmale spricht man auch von der *Wertsemantik*, die Fachwerte besitzen. Ein Fachwert repräsentiert Werte im Anwendungsbereich. Zu einem Fachwert existiert eine definierte endliche oder unendliche Wertemenge mit festgelegten Operationen[60].
Beispiele für Fachwerte sind z.B. D-Mark-Beträge, IDs, Kontonummern usw. Wie man an den D-Mark-Beträgen erkennen kann, können Fachwerte auch aus mehreren Fachwerten – z.B. Mark und Pfennig – zusammengesetzt sein. Theoretisch kann ein Fachwert beliebig komplex sein.
Zur Konstruktion von Fachwerten werden im WAM-Ansatz konkrete Entwurfsmuster verwendet[61] – z.B. das Fliegengewicht-Muster[62], um große Mengen von gleichen Werten effizient zu handhaben. Die generischen Anteile der Konstruktion von Fachwerten finden sich im JWAM-Framework wieder[63].

Mit den Entwurfsmetaphern Werkzeug, Automat, Material und Behälter sind die Arbeitsmittel und -gegenstände beschrieben, die ein Benutzer für seine Arbeit benötigt. In dem vorgestellten Satz an Entwurfsmetaphern fehlt noch der Ort, an dem sie bereitgestellt, angeordnet und aufbewahrt werden. Dies leistet die Entwurfsmetapher *Arbeitsumgebung*, die in dieser Arbeit an anderer Stelle genauer betrachtet werden wird[64].

[58] [WAM98], S.125
[59] nach [WAM98], S.119
[60] [WAM98], S.130
[61] [WAM98], S.647
[62] [Gamma97], S,199
[63] siehe package de.jwam.lang.domainvalue
[64] die Definition nach [WAM98] findet sich in Kap.3.1, S.69

1.2 Kooperationsunterstützung und CSCW

Die Kooperation mit Mitarbeitern, Dienstleistern oder Kunden ist meist wesentlicher und selbstverständlicher Bestandteil qualifizierter Arbeit. Dafür finden sich vielfältige Beispiele: eine Abteilungsleiterin delegiert eine Aufgabe an ihre Mitarbeiterinnen, ein Supportmitarbeiter berät einen Kunden, die Mitglieder eines Teams stimmen sich über die Verteilung der Aufgaben ab und bedienen sich derselben Arbeitsmittel usw.
Kooperative Arbeit tritt immer dann auf, wenn Personen bei der Erledigung ihrer Arbeit gegenseitig voneinander abhängen[65].

Entsprechend weit kann der Begriff der kooperativen Arbeit gefasst werden:

Begriff 8: Kooperative Arbeit

"Bei kooperativer Arbeit arbeiten verschiedene Personen geplant und koordiniert zusammen, um ein gemeinsames Ergebnis zu erreichen."[66]

Die Gründe für kooperative Arbeit sind sehr vielfältig und im jeweiligen Arbeitskontext zu sehen. Oft sind Aufgaben zu komplex, um von einem einzelnen Mitarbeiter bewältigt werden zu können. Oder die benötigten Ressourcen sind begrenzt, so dass sie von mehreren Mitarbeitern gemeinsam genutzt werden müssen.

Gemeinsam ist allen Formen kooperativer Arbeit, dass die beteiligten Kooperationspartner ein gemeinsames Ergebnis anstreben. Dazu ist es notwendig, dass sie sich über dieses gemeinsame Ziel verständigen und ihre Arbeitshandlungen aufeinander abstimmen. In diesem Sinne gehen die Kooperationspartner geplant und koordiniert vor. Neben festen Regeln sind es vor allem allgemein akzeptierte Konventionen, die die Koordination prägen.

Begriff 9: Koordination

"Koordination ist der Prozeß oder der Mechanismus zur Abstimmung von Arbeitsteilung bei kooperativer Arbeit. Koordination kann auf wechselseitigen Konventionen oder ausdrücklichen Regeln beruhen."[67]

Aufgaben werden zunehmend am Computer bearbeitet. Kooperationsunterstützung durch den Computer wird dadurch immer wichtiger. Oft ermöglicht der Computer erst die Kooperation zwischen Partnern, z.B. über das Internet.

Dabei werden von den Benutzern Kooperationsmittel und -medien benötigt.

[65] nach Schmidt and Bannon, 1992 in [Mark97], S.254
[66] [WAM98], S.871
[67] [WAM98], S.872

Begriff 10: Kooperationsmittel

"Ein Kooperationsmittel ist ein fachlich motivierter Gegenstand, der die Kooperation unterstützt. Er vergegenständlicht die Kooperation oder die dabei notwendige Koordination."[68]

Beispiele für Kooperationsmittel sind Vorgangsmappen und Laufzettel. Eine Vorgangsmappe ermöglicht, Unterlagen in einem arbeitsteiligen Prozess weiterzugeben. Laufzettel vergegenständlichen die Reihenfolge von kooperativen Arbeitsschritten und geben die jeweils verantwortlichen Personen an.

Begriff 11: Kooperationsmedium

"Ein Kooperationsmedium ist ein fachlich motivierter Gegenstand, der zur Realisierung von Kooperation in Anwendungssystemen dient. Gemeinsam ist allen Kooperationsmedien, daß mit ihrer Hilfe Materialien oder Information ausgetauscht werden können und daß sie selbst vergegenständlicht sind."[69]

Beispiele für Kooperationsmedien sind ein elektronisches Postversandsystem, Gruppenpostfächer oder ein elektronisches Notizbrett.

Viele computergestützte Lösungen sind jedoch technikinduziert, d.h. die technischen Möglichkeiten sind zwar vorhanden, jedoch fehlt ein intuitiv verständliches Benutzungsmodell, das die Technologien in einen fachlichen Zusammenhang stellt. Zudem stellt die fehlende Kompatibilität von Medien, Computersystemen und Applikationen oft eine schwer überbrückbare Grenze beim Austausch von Informationen bzw. Arbeitsmaterialien dar. In diesem Fall kann der Computereinsatz die Möglichkeiten zur Kooperation sogar einschränken.

In der Anwendungssoftware muss explizit vorgesehen werden, was ohne Computerunterstützung sehr informell und selbstverständlich ablaufen kann. Es bleibt also vorrangig ein anwendungsfachliches Problem, die geeigneten Formen der Computerunterstützung bei der kooperativen Arbeit zu finden[70].

Nach Lösungen für dieses Problem sucht der CSCW-Forschungszweig. Der Begriff CSCW (Computer-Supported Cooperative Work) wurde 1984 durch Irene Greif und Paul Cashman mit dem ersten Workshop zu "Computer-supported cooperative work (CSCW)" am MIT, Cambridge, eingeführt[71]. Seit 1986 finden alle zwei Jahre CSCW-Konferenzen der ACM statt. Wenig später gab es erste Konferenzen in Europa: ECSCW seit 1989, DCSCW seit 1991

[68] [WAM98], S.873
[69] [WAM98], S.874
[70] [WAM98], S.870
[71] [Teufel95], S.17

(GI, German Chapter of the ACM) – auch diese Konferenzen finden im Abstand von zwei Jahren statt.

CSCW gilt als ein junges, aber etabliertes Forschungsgebiet. Bereits an der ersten Konferenz (1984) waren nicht nur Informatiker, sondern auch Arbeitswissenschaftler, Verhaltensforscher, Psychologen, Soziologen usw. maßgeblich beteiligt. Aus der Beteiligung unterschiedlichster Forschungsdisziplinen bezieht CSCW einen Großteil seiner Dynamik und ermöglicht die Nutzung wichtiger Synergien[72].

Dementsprechend werden in CSCW viele unterschiedliche Schwerpunkte gesetzt. Bannon und Schmidt unterscheiden zwei Hauptbestandteile der Forschungsaktivität, nämlich *Computer-Support* (CS) und *Cooperative Work* (CW)[73]. Die Computerunterstützung steht also nicht unmittelbar im Mittelpunkt der Betrachtung. Das Verstehen von Gruppenaktivitäten und von deren Kontext spielt eine mindestens ebenso wichtige Rolle – nicht zuletzt, da z.B. bestehende Kooperatioonsprozesse durch den Computereinsatz nicht gestört oder gar verhindert werden sollen. Bannon und Schmidt bringen dies auf die folgende Formel: "The focus is to *understand*, so as to *better support*, cooperative work."[74].

Ziele der CSCW-Forschung sind, neue effizientere Methoden und Techniken gruppenorientierter Arbeitsweisen zu untersuchen, diese in neuen Werkzeugen zur Unterstützung der Gruppenarbeit zu implementieren und auf ihre Effektivität und Effizienz im realen Einsatz zu überprüfen[75].

Dabei muss CSCW die folgenden spezifischen Bedingungen der kooperativen Arbeit ansprechen – dies sind nach Ansicht von Bannon und Schmidt die Kernthemen des CSCW-Feldes:

" • articulating cooperative work;
• sharing an information space;
• adapting the technology to the organization, and *vice versa*."[76]

Eine allgemein anerkannte Definition von CSCW wurde jedoch noch nicht gefunden. Die Gründe werden im geringen Alter der Forschungsdisziplin und seiner interdisziplinären Natur gesehen[77]. Insbesondere der Begriff der kooperativen Arbeit ist umstritten[78] (in dieser Arbeit soll der Begriff in der Definition von [WAM98] (s.o.) verstanden werden).

[72] [Teufel95] S.17ff.
[73] [BannonSchmidt89], S.2
[74] [BannonSchmidt89], S.2
[75] [Teufel95], S.4
[76] [BannonSchmidt89], S.5
[77] [Teufel95], S.14
[78] [BannonSchmidt89], S.3

Inzwischen existieren zahlreiche kommerzielle CSCW-Applikationen, die häufig unter dem Schlagwort *Groupware* geführt werden. Nach Ellis et al. bezeichnet Groupware:

> "[...] computerbased systems, that support groups of people engaged in a common task (or goal) and that provide an interface to a shared environment."[79]

Der Begriff Groupware rückt die Technologie stärker in den Mittelpunkt und beschränkt diese zudem auf die Unterstützung kleiner Gruppen[80].

Typische Arten von Groupware-Systemen sind z.B.:

- E-Mail,
- verteilte Hypertext-Systeme,
- Shared Blackboard,
- Decision Support, Meeting facilitation,
- Desktop Conferencing,
- Message-Conferencing,
- Scheduler,
- Collaborative Writing,
- WfMS[81].

[79] Ellis et al. (1991) in [Teufel95], S.22
[80] [Grudin94], S.2
[81] nach: [Teufel95], S.1 und S.25, Böhm u.a. in [Rolf98], S.135, [Grudin94], S.9

1.3 Räumliche Konzepte in der CSCW-Diskussion und in Groupware-Systemen

In diesem Unterkapitel möchte ich auf das Verständnis der Begriffe "Ort" und "Platz" hinleiten, das dieser Arbeit zugrundeliegt. Gleichzeitig möchte ich an drei Beispielen zeigen, wie in konkreten Softwaresystemen räumliche Konzepte umgesetzt werden können.

Bei den drei vorgestellten Softwaresystemen handelt es sich um:
- den *Orbit-Mercury* Prototypen der Universität Queensland, Australien,
- das *TeamWave*-System, dessen Ursprünge an der Universität Calgary, Kanada, liegen, und
- den *POLITeam Awareness Client (POLIAwaC)* des deutschen POLITeam-Projektes.

Der Orbit-Mercury Prototyp unterstützt eine Vorstellung von Orten. Die beiden anderen Systeme unterstützen eine Vorstellung von Plätzen. Anhand der konkreten Realisierung der drei Systeme lassen sich bereits Unterschiede in der Bedeutung der beiden Begriffe ahnen.

Diese Systeme dienen in vielen Teilen als Vorbild für den Entwurf der COJAC-Raumkomponente.

In allen drei vorgestellten Systemen spielt die Unterstützung von *Awareness* eine Rolle. Daher möchte ich diesen Begriff hier im Vorfeld definieren.

Definitionen des Begriffs "Awareness" sind in der Literatur zahlreich vorhanden. Sie reichen von sehr pauschalen Beschreibungen, wie "knowing what is going on", bis hin zur komplexen Ausdifferenzierung in mehrere Arten von Awareness[82].
Die folgende Definition des Begriffs liegt dazwischen und beruht insbesondere auf den Definitionen von Dourish und Bellotti[83] und von Roock und Wolf[84].

Begriff 12: Awareness

Awareness ist das Verständnis der Kooperationssituationen, in die ein Benutzer involviert ist. Grundlage dafür ist die gegenseitige Wahrnehmung der Kooperationspartner und ihrer Handlungen.

Zudem wird der Begriff "Awareness" gelegentlich in zwei weiteren Bedeutungen verwendet: für die Unterstützung von Awareness in einem Softwaresystem und für die Awarenessinformationen selbst.

Awareness ermöglicht Benutzern, sich zu koordinieren und ihre Arbeit zu strukturieren. Ohne Awareness ist kooperative Arbeit praktisch unmöglich. Mit Awareness sind jedoch zwei Probleme verbunden: zum einen die Störung des Arbeitsflusses durch Informationsüberflutung und zum anderen die mögliche Verletzungen der Privatsphäre[85].

[82] siehe z.B. jeweils [GutwinGreenberg99], S.8
[83] 1992, in [Berlage99], S.207
[84] [RoockWolf98], Anforderung 5, S.67 und [WAM98], S.931
[85] [Sohlenkamp98], S.140f.

Awarenessinformationen können auf sehr unterschiedliche Weise präsentiert werden. Dabei gilt: Je stärker ein Benutzer durch die Präsentation gestört wird, desto stärker nimmt er sie wahr. Die Reihe der möglichen Präsentationsweisen von Awarenessinformationen reicht dabei von überhaupt keiner Benachrichtigung über unauffällige Anzeigen im Hintergrund, Animationen im Hintergrund, nicht-modalen Fenstern bis hin zu modalen Dialogboxen. Letztere erfordern vom benachrichtigten Benutzer eine Interaktion. Auf diese Weise werden sie sicher beachtet, stören den Benutzer in seiner Arbeit aber besonders stark[86].

Im Folgenden stelle ich nun zunächst die drei Systeme vor und beschreibe damit verbundene Konzepte sowie deren Handhabung und Präsentation. Anschließend vergleiche ich die Systeme hinsichtlich der konkreten Umsetzung der Konzepte.

1.3.1 Das locales framework und der Orbit-Mercury-Prototyp

Das *wOrlds-Projekt* am Cooperative research Centre for Distributed Systems Technology (DSTC) der Universität Queensland, Australien, entwickelt parallel eine Theorie zur kooperativen Arbeit, das *locales framework*, und eine Serie von drei Prototypen, *Orbit-Mercury*, *Orbit-Lite* und *Orbit-Gold*, um diese Theorie zu prüfen[87].

Das locales framework ist als Werkzeug gedacht, das Designern helfen soll, die Mittel zu identifizieren, die in einem bestimmten Kontext zur Kooperationsunterstützung benötigt werden[88].

Vielen Groupware-Systemen liegt ein Verständnis von Orten (*locales*) zugrunde[89]. Das locales framework verfolgt das Ziel, die gemeinsamen Aspekte dieser Systeme zusammenzufassen und zu benennen[90].

[86] [Berlage99], S.229. Im weiter unten beschriebenen POLITeam Awareness Client (Kap1.3.3, S.34) kann dies der Benutzer einstellen, indem er einem Awarenessprofil eine Intensitätsstufe (0-4) zuweist, die das System dann auf bestimmte Präsentationsweisen abbildet ([Sohlenkamp98], S.145).
[87] [Mansfield98], S.1
[88] [Mansfield98], S.2 – dabei sind nicht nur Softwaredesigner gemeint, da die Konstruktion physischer, virtueller und hybrider Orte gleichermaßen miteinbezogen wird
[89] wie z.B. den hier beschriebenen Systemen, aber auch z.B. Webseiten mit FAQ-Listen (letztere sowie den Vorgänger des TeamWave-Systems, TeamRooms, führt [Mansfield98] auf S.4 als Beispiele an)
[90] Die Aspekte beinhalten keine konkreten Lösungen für die mit diesen Aspekten verbundenen Probleme.

Knapp formuliert lassen sich diese Aspekte wie folgt beschreiben:

- *locale foundations* – Das grundlegende Ortsverständnis. Ein Ort (locale) ist ein "Brennpunkt", um den sich die für bestimmte kooperative Tätigkeiten relevanten Personen, Objekte, Werkzeuge und Ressourcen versammeln.
- *civic structures* – Die Einbettung eines Ortes in seinen sozialen Kontext
- *individual view* – Die verschiedenen individuelle Sichten unterschiedlicher Personen auf denselben Ort und die individuelle Sicht einer Person auf mehrere Orte.
- *trajectory* – Zeitliche Aspekte der Interaktion.
- *mutuality* – Präsenz und Awareness[91].

Um das locales framework zu testen, sind die drei separat entwickelten Orbit-Prototypen entstanden. Ich beschreibe hier den ersten dieser drei, Orbit-Mercury, der m.E. die interessantesten Merkmale aufweist.

Alle drei Orbit-Prototypen sind in drei Schichten aufgebaut, in denen sich Aspekte des locales Framework widerspiegeln (siehe Abb.2).

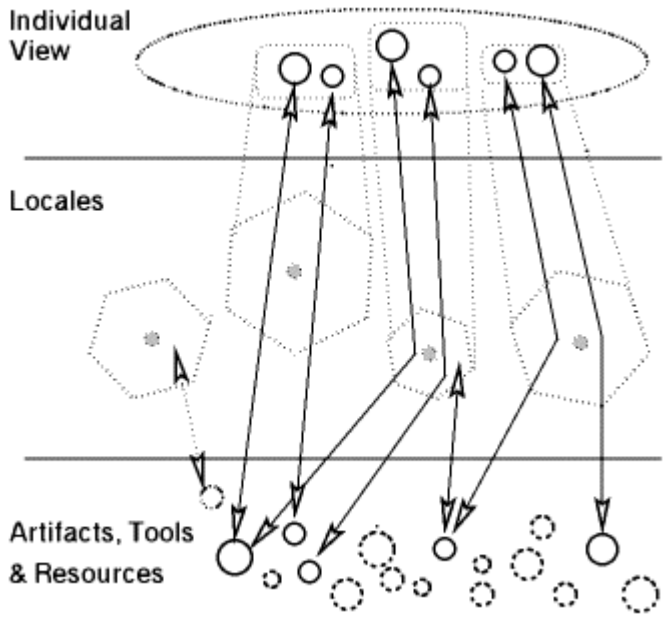


Abbildung 2: Das *three-layer model* der Orbit-Prototypen[92]

In der untersten Schicht befinden sich die gemeinsam genutzen Objekte. Diese werden gemäß ihrer Verwendung durch Gruppen zu Orten zusammengefasst, die sich wiederum zu der individuellen Sicht eines Benutzers zusammenfügen[93].

Dem Ort-Verständnis liegt hier scheinbar keine Vorstellung der Einheit von Raum und Zeit zugrunde, da sich – wie die Abb.2 zeigt – Objekte an mehreren Orten gleichzeitig befinden

[91] [Mansfield98], S.2ff.
[92] [Mansfield98], Abb.1, S.5
[93] [Mansfield98], S.5

können. Die Einheit von Raum und Zeit wird allerdings auch nicht ausgeschlossen, zumal Orte nicht notwendigerweise als Behälter verstanden werden und sich teilweise überlappen können.

Die Orbit-Prototypen implementieren die oberen beiden Schichten. Die oberste bilden die sogenannten *Orbit-Clients*, die mittlere ein sogenannter *locales service* und die unterste Schicht schließlich sogenannte r*epositories*, z.B. File-Systeme, das Web, USENET-News, Datenbanken usw[94]. Die repositories, der locale service sowie die Orbit-Clients können alle auf verschiedenen Rechnern laufen.

Die Absicht ist, mit Orbit einen *ubiquitous collaborative desktop* anzubieten, über den Benutzer alle kooperativen und individuellen Aufgaben erledigen. Eine wichtige Eigenschaft des Entwurfs ist dabei jedoch, dass der Orbit-Client lediglich für die Visualisierung der Objekte an den jeweiligen Orten zuständig ist. Er dient nicht unmittelbar dem Bearbeiten von Objekten und Ressourcen. Dazu öffnet er jeweils ein spezielles, geeignetes Werkzeug[95].

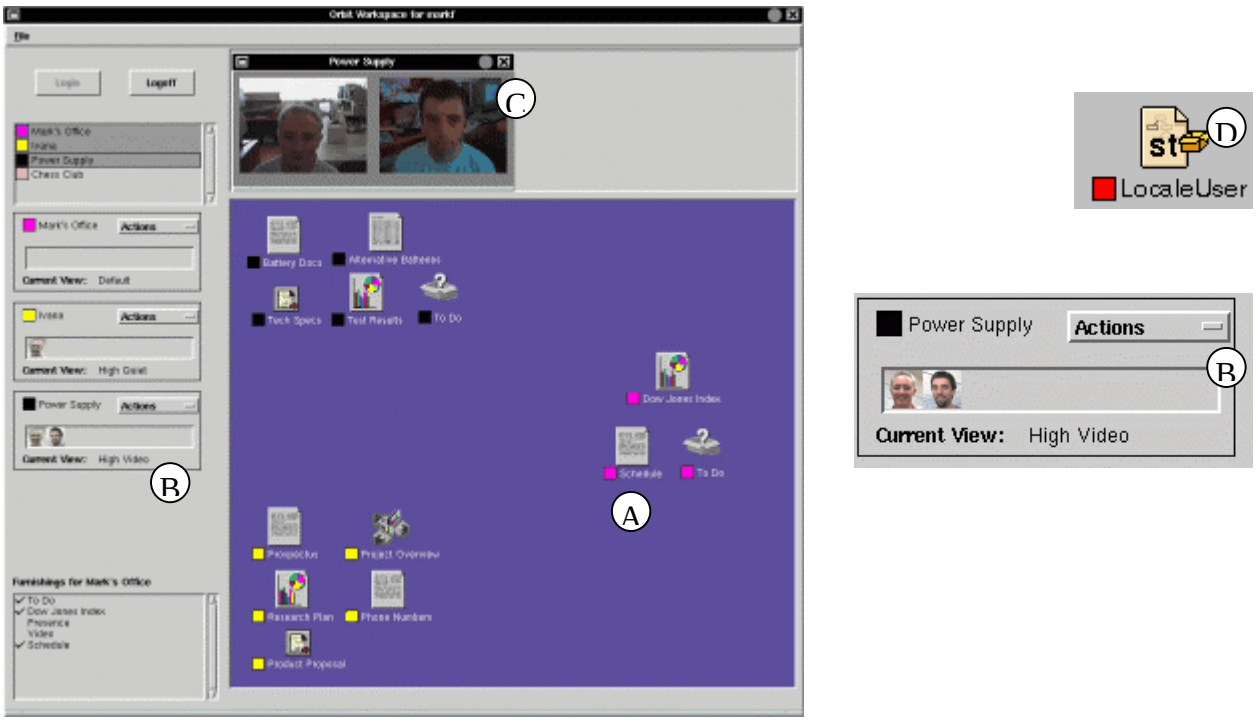


Abbildung 3: Der Orbit-Mercury-Prototyp, Details[96]

Auf dem Orbit-Desktop (siehe Abb.3) befinden sich Icons, die die Objekte an den Orten symbolisieren und die je nach Ortszugehörigkeit mit einer Farbe gekennzeichnet werden (A).
Der Desktop ist also nicht – wie man vielleicht erwartet hätte – in Bereiche aufgeteilt, die den visualisierten Orten entsprechen. Stattdessen können die Objekte ungeachtet ihrer Ortszugehörigkeit beliebig auf dem Desktop angeordnet werden[97].

---

[94] [Mansfield98], S.6
[95] [Mansfield98], S.5f.
[96] [Mansfield98], Abb.5, S.10 und Abb.3+4, S.8
[97] In der Abb.3 sind die Icons "zufälligerweise" ihrer Ortszugehörigkeit nach gruppiert.

Die individuelle Sicht eines Benutzers kann unter einem Namen gespeichert und wahlweise mit anderen Benutzern geteilt werden. Änderungen an den Positionen der Objekte wirken sich dann für alle Benutzer aus, die dieselbe Sicht teilen.
Für jeden Ort wird links neben der Desktopfläche angezeigt, welche Benutzer diesen Ort sehen können und welche individuelle Sicht sie darauf eingestellt haben (B)[98].

Abhängig von der eingestellten Sicht auf einen Ort bietet Orbit eine einfache Audio-Video-Verbindung zwischen den Benutzern mit der gleichen eingestellten Sicht an (C). Auf diese Weise sollen Benutzer ohne große Mühen mit anderen Benutzern explizit kommunizieren können. Dabei ist es möglich, über die Audio-Video-Verbindung verschiedener Orte mehrere Gespräche parallel zu führen[99].

An den Objekt-Icons wird über kleinere Ereignis-Icons visualisiert, welches Ereignis zuletzt an dem entsprechenden Objekt eingetreten ist. In der Abb.3 symbolisiert z.B. ein geöffneter Karton, dass das dekorierte Objekt zuletzt von einem Benutzer geöffnet wurde (D)[100].
Tritt ein Ereignis an einem Objekt ein, so werden die Desktops aller Benutzer automatisch aktualisiert[101].
Darüberhinaus werden Benutzer synchron informiert, wenn bestimmte Ereignisse eintreten, wie z.B. das Erzeugen neuer Orte[102].

1.3.2 Das TeamWave-System

TeamWave ist ein kommerzielles Internet-Groupware-Produkt, das auf einer Raummetapher basiert. Es ist die Fortentwicklung eines zu Forschungszwecken entwickelten Prototypen namens *TeamRooms*, der im Rahmen des *Grouplab-Projekts* an der Universität Calgary, Kanada, entwickelt wurde. Die hier vorgestellte Version von 1997 ähnelt diesem Prototypen stark[103].

Die Entwicklung von TeamWave zielt vor allem darauf ab, den Übergang zwischen verschiedenen Arbeitsstilen zu erleichtern.
Viele Groupware-Systeme enthalten Hindernisse für die natürliche soziale Interaktion oder sie lassen Benutzer nicht ohne weiteres zwischen verschiedenen Arbeitsstilen wechseln[104].

Die CSCW-Taxonomie, wie sie z.B. durch die Time-Space-Matrix[105] ausgedrückt wird, spiegelt nicht wider, auf welche Weise Personen in der Praxis zusammenarbeiten. Viele

[98] [Mansfield98], S.7f.
[99] [Mansfield98], S.9
[100] [Mansfield98], S.8
[101] [Mansfield98], S.9
[102] [Mansfield98], S.10
[103] [GreenbergRoseman98], S.7. Die Homepage der Firma TeamWave Software Ltd., die Produkt entwickelt, findet sich unter: http://www.teamwave.com/. Ein besonderer Kundenkreis für das TeamWave-System geht aus den entsprechenden Broschüren, soweit ich erkennen konnte, nicht hervor.
[104] [GreenbergRoseman98], S.1
[105] [Grudin94], Abb.3, S.9

Groupware-Applikationen lassen sich genau einer Zelle der Time-Space-Matrix zuordnen. Demzufolge müssen Benutzer je nach Kooperationstil zwischen den entsprechenden Groupware-Applikationen wechseln. Dies behindert nach Ansicht von Greenberg und Roseman die Benutzer darin, die räumlichen, zeitlichen und funktionalen Grenzen bei der kooperativen Arbeit zu überschreiten[106].

Daneben existieren zahlreiche weitere Grenzen in Groupware, z.B. zwischen individueller und kooperativer Arbeit, Technologien, formalen und informalen Handlungen usw.[107]

Um diese Grenzen zu entschärfen oder sogar zu überwinden, schlagen Greenberg und Roseman vor, Systeme nach der Raummetapher zu gestalten. Diese unterstützen ein weites Sortiment an Kooperationsstilen innerhalb einer einzelnen Umgebung[108].

Sie entwickeln eine (unvollständige) Liste von Merkmalen, die mit der Raummetapher verbunden sind – dabei beziehen sie sich auf die Merkmale einer räumlichen Umgebung nach Harrison und Dourish[109]:

- Ein Raum ist ein begrenzter Ort[110].
- Ein Raum ist ein persistenter Behälter von Objekten, der von Personen an ihre Bedürfnisse angepasst werden kann.
- In einem Raum sind Objekte räumlich angeordnet[111].
- Ein Raum ist "bevölkert"[112].

Obwohl diese Merkmale selbstverständlich erscheinen, werden sie von den wenigsten Groupware-Systemen unterstützt[113] – anders das Teamwave-System, das ich im folgenden vorstellen möchte.

Die Abb.4 auf der nächsten Seite zeigt die TeamWave Workplace Benutzerschnittstelle, in der ein Raum visualisiert wird und weitere Fenster zu sehen sind.

[106] [GreenbergRoseman98], S.2
[107] [GreenbergRoseman98], S.3
[108] [GreenbergRoseman98], S.4 Der Begriff "Raum" der Raum-Metapher steht für den englischen Begriff "room" im Sinne von "Zimmer".
[109] [GreenbergRoseman98], S.4ff. – zu Harrison und Dourish: Kap.1.3.5, S.39
[110] *partitioning* bei Harrison und Dourish
[111] bei Harrison und Dourish sowohl *relational orientation and reciprocity* als auch *proximity and action*
[112] *presence and awareness* bei Harrison und Dourish
[113] [GreenbergRoseman98], S.7

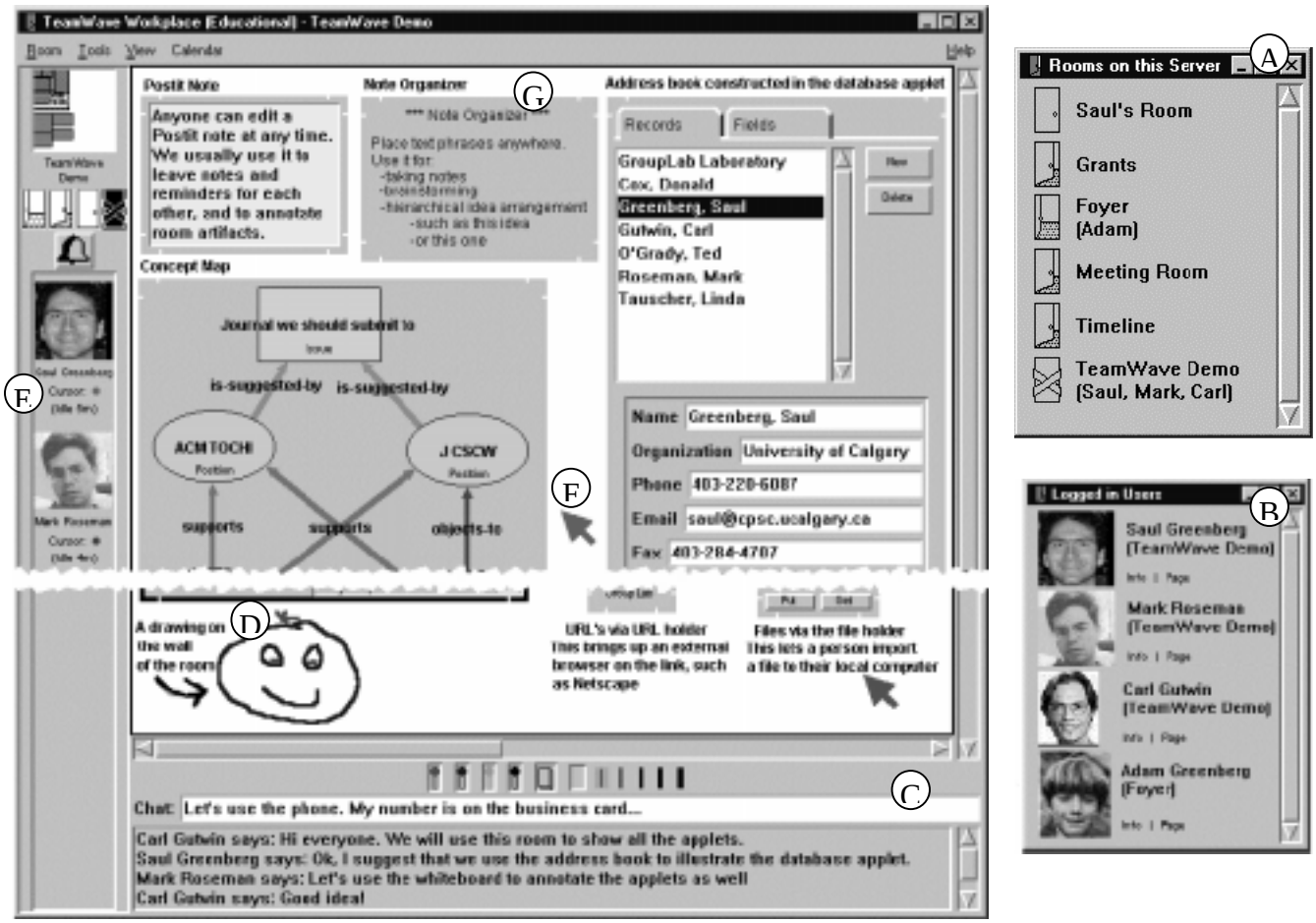


Abbildung 4: Das TeamWave-System, Werkzeuge[114]

TeamWave verwendet eine Client-Server-Architektur, wobei sich die gemeinsam benutzten Räume auf dem Server befinden. Verbindet ein Benutzer seinen Client mit dem Server, so wird automatisch ein (defaultmäßiger) Raum angezeigt[115].

Über ein spezielles Fenster kann der Benutzer einen anderen Raum auswählen (A). Zu jedem Raum werden dort die präsenten Benutzer angezeigt. Die Türsymbole in dem Fenster visualisieren den momentanen Grad der Privatheit eines Raumes von öffentlich über zwei "Bitte nicht stören"-Grade bis privat[116].

In einem weiteren Fenster sind die eingeloggten Benutzer und der Raum, in dem sie präsent sind, sichtbar (B)[117].
Räume können von Benutzern neu erzeugt und – sofern sie die Rechte dazu besitzen – gelöscht werden[118].

[114] nach [GreenbergRoseman98], Abb.1, S.8
[115] [GreenbergRoseman98], S.7
[116] [GreenbergRoseman98], S.7
[117] [GreenbergRoseman98], S.10
[118] [GreenbergRoseman98], S.10

Über den Versand von Nachrichten und über ein einfaches Chat-Tool, das in jedem Raum stets zur Verfügung steht, können Benutzer explizit kommunizieren (C). Zudem ist die Hintergrundfläche eines visualisierten Raumes (die sog. "Wand") eine Art Schreibtafel, auf der Benutzer Freihandzeichnungen und Text hinterlassen können (D)[119].

Awarenessinformationen werden raumbezogen angezeigt. In jedem Raum werden die präsenten Benutzer visualisiert (E). Über farblich unterschiedliche *Telepointer* sind die Mauszeiger aller Benutzer für jeden, der in einem Raum präsent ist, sichtbar (F).
Gemeinsam benutzte Räume stellen sich für alle Benutzer synchron auf die gleiche Weise dar. Lediglich der sichtbare Ausschnitt kann von einem Benutzer individuell gewählt werden[120].

In jedem Raum stehen Werkzeuge zur Verfügung. Diese Werkzeuge sind Groupware-Applikationen für spezielle Zwecke, die die Funktionalität eines Raumes erweitern. Aktivierte Werkzeuge werden in einer Art Applet innerhalb des Raumes für alle Benutzer gleichermaßen visualisiert (G)[121].

Die in TeamWave umgesetzte Raummetapher erleichtert den Übergang vor allem zwischen bestimmten Arbeits- und Kooperationsstilen:

- Übergang zwischen individueller und kooperativer Arbeit:
  Greenberg und Roseman betrachten individuelle Arbeit als *reflexive CSCW*: Ein Benutzer koordiniert sich und kommuniziert asynchron mit seinem "zukünftigen Selbst". Individuelle Arbeit ist insofern auch kooperative Arbeit.
  Zudem besteht in TeamWave kein Unterschied in der technischen Realisierung von privaten und öffentlichen Räumen. Indem individuelle Arbeit in den gemeinsam verwendbaren Räumen verrichtet wird, besteht für einen Benutzer die Möglichkeit, problemlos die individuelle Arbeit zu unterbrechen und mit anderen Benutzern zu interagieren. Sobald ein anderer Benutzer den Raum eines einzelnen Benutzers auswählt, wird dieser Einzelraum zum Gruppenraum[122].
- Asynchrone und synchrone Kooperation:
  Durch ihre synchrone Präsentation ermöglichen die TeamWave-Räume stets die synchrone Kooperation. Durch die Anzeige von Awarenessinformationen, wie z.B. den in einem Raum präsenten Benutzern, regen sie zudem die explizite Kommunikation an.
  Durch ihre persistente Natur gestatten die TeamWave-Räume desgleichen die asynchrone Kooperation, indem Arbeitsgegenstände bereitgelegt, Markierungen angebracht und Bemerkungen hinterlassen werden können[123].

1.3.3 POLIAwaC, der POLITeam Awareness Client

Das *POLITeam-Projekt*[124] wurde anlässlich des 1991 beschlossenen Umzugs der Bundesregierung von Bonn nach Berlin ins Leben gerufen. Einige Ministerien und Behörden

[119] [GreenbergRoseman98], S.10 und S.19
[120] [GreenbergRoseman98], S.10
[121] [GreenbergRoseman98], S.11
[122] [GreenbergRoseman98], S.12ff.
[123] [GreenbergRoseman98], S.15 u. S.18

behalten ihren Sitz vorerst in Bonn. Ziel des POLITeam-Projektes ist, für diese verteilten Ministerien und Behörden Möglichkeiten zur Telekooperation zu entwickeln.
Dazu sollen Arbeitsprozesse, die auf schriftlichen Dokumentieren basieren, durch elektronische ergänzt werden. Als wichtigste kooperationsunterstützenden Konzepte verwendet das Anfang 1995 installierte *POLITeam-System* dazu elektronische Umlaufmappen[125] und gemeinsame Arbeitsbereiche[126].

Arbeitsbereiche sind in POLITeam Behälter, die andere Objekte wie z.B. Dokumente, Werkzeuge oder auch weitere Arbeitsbereiche enthalten können[127].

Die gemeinsamen Arbeitsbereiche sind soziale Umgebungen, in der die Handlungen eines Einzelnen die übrigen Personen in ihrer Arbeit beinflussen. Ein wichtiges Mittel, um die unterschiedlichen Sichtweisen und Arbeitsstile verschiedener Personen zusammenzuführen, sind Konventionen. Koventionen regeln das Verhalten in der Gruppe und betreffen die Verfahrensweisen bei der Systembenutzung[128].

Die Rolle von Konventionen wird am Groupware-System POLITeam seit 1996 unter realen Bedingungen untersucht[129]. Als Ergebnisse konnte dabei festgehalten werden:

- Konventionen sind dynamisch und können unvorhersehbar sein. Sie entstehen aus der Arbeit heraus.
- Damit Konventionen definiert werden können, ist es notwendig, dass die Arbeitspraktiken der anderen Gruppenmitglieder verstanden werden
- Konventionen scheitern, wenn das Verhältnis zwischen Kosten und Nutzen unausgewogen ist. Selbst wenn die Einhaltung solcher Konventionen durch das System erleichtert wird, finden die Benutzer Wege, sie zu umgehen[130].

Aufgrund dieser Ergebnisse wird im POLITeam-System der Ansatz verfolgt, Konventionen durch implizite, nicht-steuernde technische Mittel zu unterstützen. Als ein solches Mittel werden Awarenessinformationen angezeigt, durch die das gegenseitige Verständnis der Benutzer für ihre Arbeitspraktiken erhöht wird. Anstatt das Verständnis von Konventionen im System zu erfassen, wird so die dynamische Bildung von Konventionen ermöglicht[131].

Im Rahmen des POLITeam Projekts wurde dazu der *POLITeam Awareness Client* (*POLIAwaC*) auf der Grundlage der Desktopmetapher realisiert. Er ergänzt die übliche Groupware-Funktionalität um Awareness-Mechanismen[132].

---

[124] Die URL der POLITeam-Homepage lautet: http://orgwis.gmd.de/projects/POLITeam/
[125] auf die ich hier nicht weiter eingehe – [Breitling00] betrachtet sie in seiner Arbeit genauer (S.56ff.)
[126] [Mark97], S.254.
[127] [Sohlenkamp98], S.142
[128] [Mark97], S.253f. Nach [HarrisonDourish86], S.69f. sind Konventionen Teil des Platzverständnisses, welches dem System prinzipiell nicht inhärent sein kann (s.u).
[129] [Mark97], S.254
[130] [Mark97], S.266f.
[131] [Mark97], S.263 und S.267
[132] [Mark97], S.263 und [Sohlenkamp98], S.140

POLIAwaC zeigt, dass es möglich ist, die Anzeige von Awarenessinformationen in ein System zu integrieren, ohne dass dessen allgemeines Erscheinungsbild dadurch verändert wird[133].

Der POLIAwaC visualisert seine hierarchisch geordneten Arbeitsbereiche in drei spezialisierten Ansichten (siehe Abb.5). Die Ansicht auf der linken Seite zeigt diese Hierarchie. In dieser Ansicht werden die Benutzer symbolisiert, die in einem Arbeitsbereich aktiv sind (A). Ein Benutzer gilt als aktiv, wenn er einen Arbeitsbereich in der Hierarchie-Übersicht geöffnet hat. Um zur selben Zeit in mehreren Arbeitsbereichen aktiv zu sein, kann ein Benutzer mehrere Fenster öffnen.
Die Ansicht im oberen Teil des Fensters beinhaltet eine desktopähnliche Darstellung der Objekte des in der linken Ansicht ausgewählten Arbeitsbereichs. Die Icons der Objekte können vom Benutzer auf dieser "Desktopfläche" beliebig angeordnet werden.
Die dritte Ansicht zeigt in Form einer Liste eine detailliertere Übersicht über dieselben Objekte.[134].

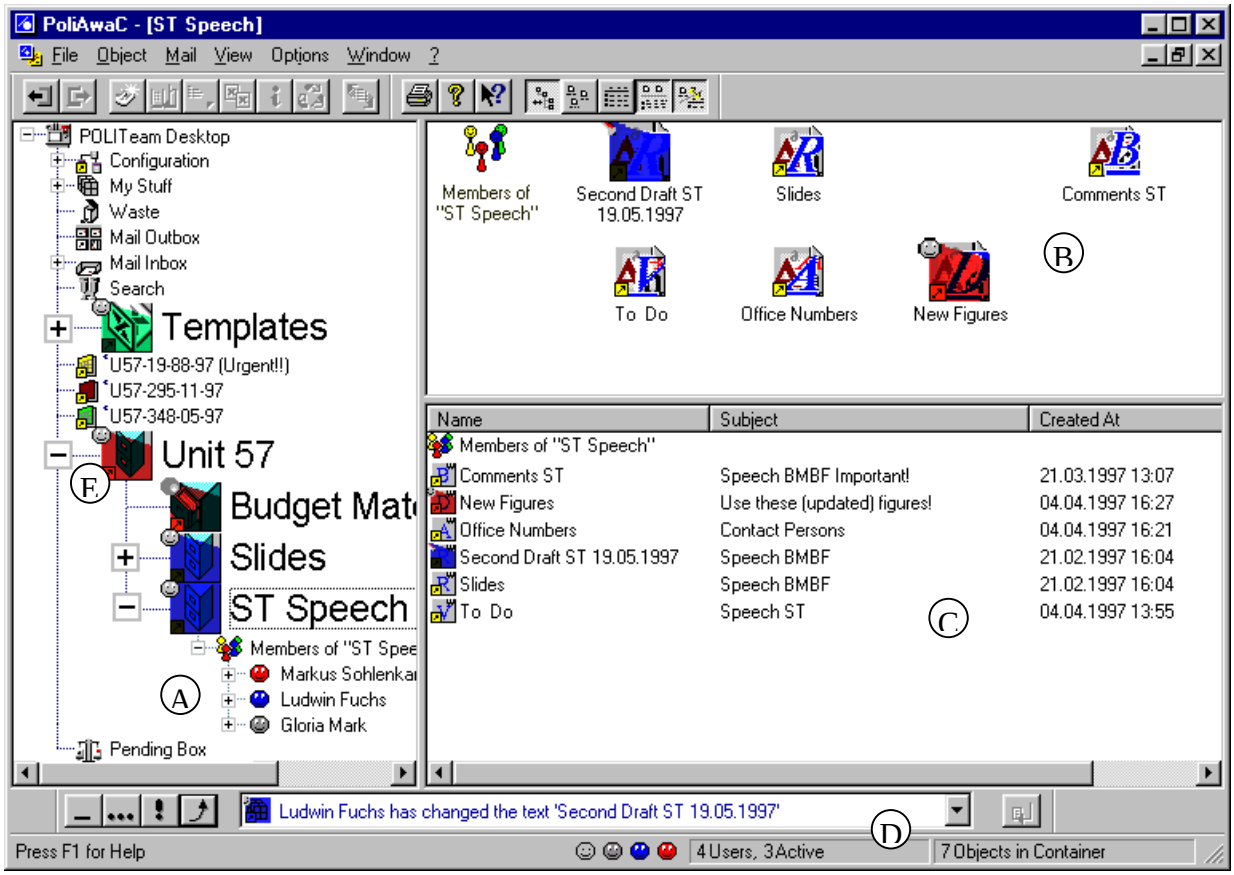


Abbildung 5: Der POLITeam Awareness Client[135]

Arbeitsbereiche können privat oder gemeinsam genutzt werden – ein wichtiges Ziel beim Entwurf des POLIAwaC war, den reibungslosen Übergang zwischen individueller und

[133] [Sohlenkamp98], 151
[134] [Sohlenkamp98], S.142f.
[135] [Mark97], Abb.1, S.264

gemeinsamer Arbeit zu ermöglichen. Arbeitsbereiche können wie die anderen Objekte neu erstellt, verschoben, kopiert werden usw.[136]

Awarenessinformationen werden als nicht-störende Hinweise angezeigt. So bedeutet z.B. ein vergrößertes Icon, dass das symbolisierte Objekt von einem Benutzer verwendet wird – zudem wird es mit einem kleinen Benutzericon dekoriert (E). Auch die aktiven Benutzer selbst werden mit diesen Icons visualisiert (A).
Neben diesen synchronen Informationen zeigt eine sogenannte Ereignisleiste asynchron eine chronologische Folge von objektbezogenen Ereignissen (D)[137].

Jedes Objekt definiert eine Reihe von Handlungen, die an ihm möglich sind. Welche davon in einer Arbeitssituation interessant sind, kann ein Benutzer über ein spezielles *Awarenessprofil* festlegen. Nur diese werden dann im gegebenen Falle als Awarenessinformation visualisiert.
Diese Awarenessprofile können gemeinsam genutzt werden. Auf diese Weise lassen sich Ereignisse, die für die Einhaltung von best. Konventionen relevant sind, gezielt berücksichtigen und weniger wichtige Ereignisse vernachlässigen.
In das Textfeld dieser Ereignis-Leiste können Benutzer Nachrichten eintragen, um so die automatisch generierten Awarenessinformationen um informale Informationen zu ergänzen (z.B. warum eine best. Konvention von einem Benutzer verletzt wurde)[138].

1.3.4 Vergleich der drei Groupware-Systeme

Alle drei Systeme visualisieren eine gemeinsame Umgebung.
Orbit-Mercury verwendet eine Vorstellung von Orten und unterstützt ein gemeinsames Verständnis derselben durch gemeinsame "virtuelle" Sichten darauf. Da verschiedene Sichten dieselben Orte umfassen können, kann mit ihnen keine Vorstellung von gemeinsamen Plätzen verknüpft werden.
TeamWave und POLIAwaC basieren dagegen auf einer Vorstellung von Plätzen. In TeamWave werden diese in Form von virtuellen Räumen konkretisiert. In POLIAwaC handelt es sich bei den Plätzen um einen gemeinsamen Desktop und darin beliebig geschachtelte Arbeitsbereiche.

In Orbit-Mercury können sich Dinge zu einem Zeitpunkt an mehreren Orten befinden. Dies widerspricht der Einheit von Raum und Zeit nicht, wenn dem die Vorstellung zugrunde liegt, dass sich Orte teilweise überlappen können: Dinge in der "Schnittmenge" befinden sich an beiden Orten. Nach meinem Eindruck ist das jedoch nicht der Fall.
TeamWave und POLIAwaC unterstützen die Einheit von Raum und Zeit, d.h. Dinge existieren zu einem Zeitpunkt nur einmal innerhalb der räumlichen Umgebung.

In Orbit-Mercury werden gemeinsame Orte nur für Benutzer, die dieselbe Sicht eingestellt haben, auf gleiche Weise visualisiert. Die Plätze in TeamWave und POLIAwaC werden

[136] [Sohlenkamp98], S.142f.
[137] [Mark97], S.264
[138] [Mark97], S.264ff.

dagegen für alle Benutzer auf die gleiche Weise dargestellt – TeamWave zeigt dabei sogar die Mauszeiger der anderen Benutzer an.

In allen drei Systemen dient eine desktop-artige Fläche der Visualisierung der Orte. In Orbit-Mercury visualisiert diese Fläche alle Dinge, die zu der eingestellten Sicht gehören – also u.U. mehrere Orte. In TeamWave und POLIAwaC enthält diese Fläche die Dinge eines einzelnen Raumes bzw. Arbeitsbereichs.
Auf den desktop-ähnlichen Flächen der drei Systeme können Dinge räumlich angeordnet und z.B. gruppiert werden.

Eine räumliche Anordnung der Orte selbst, z.B. welche Orte nah beieinander liegen, spielt in keinem der drei Systeme eine Rolle.

Die prinzipiellen Zugriffsmöglichkeiten auf die Dinge an den gemeinsamen Orten – d.h. ungeachtet eines möglichen Zugriffsschutzes – sind in allen drei Systemen für alle Benutzer gleich.
Für den Orbit-Prototypen ist Zugriffsschutz noch Gegenstand der Forschung und existiert nicht. Für die TeamWave-Räume können Zugriffsrechte explizit festgelegt werden. Für POLIAwaC wird eine Möglichkeit des expliziten Zugriffsschutzes als unnötig angesehen. Dies wird mit der Beobachtung begründet, dass Benutzer private Arbeitsbereiche i.a. respektieren[139].

Alle drei Systeme bieten die Möglichkeit, Dinge innerhalb der gemeinsamen Umgebung zu bearbeiten.
Unklar ist geblieben, in welchem Zusammenhang die drei Systeme jeweils zur Unterstützung individueller Arbeit durch einen Einzelarbeitsplatz stehen.
TeamWave bietet die Möglichkeit, "non-TeamWave objects" (z.B. Dokumente einer externen Textverarbeitung) als Datei in die TeamWave-Räume hochzuladen. Desgleichen können diese Dateien auf einen Einzelarbeitsplatz heruntergeladen und dort, außerhalb von TeamWave, bearbeitet werden[140]. Diese Funktionalität scheint jedoch eher auf die Einbindung anderer Systeme abzuzielen, als auf die Unterstützung individueller Arbeit.

Alle drei Systeme unterstützen Awareness. So werden die Benutzer über bestimmte Ereignisse informiert, die sich auf die Objekte in der gemeinsamen Umgebung beziehen. Zudem visualisieren alle drei Systeme, welche Benutzer einen Ort sehen können.
Awarenessinformationen werden jeweils auf nicht-störende Weise z.B. als dekorierte Icons oder Benutzersymbole angezeigt.

Alle drei Systeme bieten eine Möglichkeit zur expliziten Kommunikation.
In Orbit-Mercury werden abhängig von der eingestellten Sicht Audio-Video-Verbindungen aufgebaut, die an bestimmte Orte geknüpft sind.

[139] [Sohlenkamp98], S.142
[140] [GreenbergRoseman98], S.12

In TeamWave besteht die Möglichkeit zum Chat. Von einer Audio- oder Video-Verbindung wurde hier abgesehen, um die Hardwareanforderungen möglichst gering zu halten[141].
POLIAwaC bietet die Möglichkeit, Textnachrichten in eine Ereignisleiste einzutragen. Dies dient im Ggs. zum Chatten jedoch eher der asynchronen Kommunikation. Darüberhinaus sieht POLIAwaC die Möglichkeit vor, Chat- oder Audio-Video-Tools einzubetten. Diese können z.B. über ein Benutzericon aktiviert werden.

1.3.5 Re-Place-ing Space

Harrison und Dourish untersuchen in ihrem Artikel kritisch, ein räumliches Verständnis (*notion of space*) als Basis für den Entwurf von CSCW-Systemen zu verwenden. Sie vertreten die Auffassung, dass die entscheidende Eigenschaft, nach der Softwaredesigner suchen – nämlich einem angemessenen Verhaltensrahmen (*appropriate behavioural framing*) – ganz und gar nicht in den Eigenschaften einer räumlichen Umgebung[142] verwurzelt ist, sondern vielmehr in einem gemeinsamen, verbreiteten kulturellen Verständnis von Plätzen (*notion of place*)[143].

Dennoch sehen sie die Bedeutung des räumlichen Umgebung für die alltägliche Arbeit.
Als für die Kooperationsunterstützung relevant benennen sie die folgenden Merkmale der räumlichen Umgebung:

- *relational orientation and reciprocity* – Das räumliche Verständnis wird allgemein geteilt.
- *proximity and action* – Handlungen und Interaktionen beziehen sich zumeist auf die Dinge und Personen in unserer Nähe.
- *partitioning* – Handlungen und Interaktionsmöglichkeiten können (z.B. durch Türen, Wände, Entfernung) voneinander abgegrenzt werden.
- *presence and awareness.*

Damit unterstützen räumliche Kooperationsmodelle das räumlich geprägte Verhalten, so wie es sich in unseren alltäglichen Handlungen zeigt[144].

Es sind jedoch Plätze, nicht Orte, die den Rahmen für angemessenes Verhalten bilden[145].
Die Unterscheidung zwischen einem Ort und einem Platz entspricht der Unterscheidung zwischen "Haus" und "Heim". Trotz gleicher räumlicher Eigenschaften kann sich das

---

[141] [[GreenbergRoseman98], S.10
[142] Der englische Begriff "space" wird von [HarrisonDourish96] im Sinne von "räumlicher Umgebung" verwendet. Ist die gesamte räumliche Umgebung an sich gemeint, übersetze ich "space" mit "räumliche Umgebung" (anstelle von "Raum", um eine Verwechslung im Sinne von "Zimmer" zu vermeiden). Ist nur eine konkrete räumliche Umgebung gemeint, verwende ich den Begriff "Ort". Aus den Beispielen, die [HarrisonDourish96] für Orte geben (z.B. ein wenig einladendes Backstein-Portal vor EuroPARC, um das sich Raucher zur Zigarettenpause versammeln u.ä.) schließe ich, dass sie Orte im Sinne des später entstandenen locales framework aus [Mansfield98] verstehen. Dort ist ein Ort ein "Brennpunkt", um den sich Personen und Objekte versammeln. (Kap.1.3.1, S.39)
[143] [HarrisonDourish96], S.67
[144] [HarrisonDourish96], S.68
[145] [HarrisonDourish96], S.73

Verständnis von Plätzen unterscheiden. So mag derselbe Ort zeitweise als ein anderer Platz aufgefasst werden[146].
Das Platzverständnis verändert den Ort. Verhalten, das dem Verhaltensrahmen eines Platzes nicht entspricht, wird als "deplaziert" betrachtet[147].

Mit ihren räumlichen Eigenschaften wie physischer Struktur, Topologie, Orientierung und Verknüpfung bieten Orte diverse Möglichkeiten und Einschränkungen[148].
Ein Platz erhält i.A. viel von seiner Bedeutung aufgrund der räumlichen Gegebenheiten, doch kommen soziale Bedeutung, Konventionen, kulturelles Verständnis von Rolle, Funktion und Natur usw. hinzu[149]. Insofern sind virtuelle Plätze Teil der realen Welt[150].

Harrison und Dourish bringen den Unterschied zwischen Ort und Platz auf die Formel:

> "Space is the opportunity; place is the understood reality."[151]

Wie Orte als Plätze verstanden werden, kann durch den Systementwurf nicht vorgegeben werden. Dieses Platzverständnis wird erst von den Benutzern erzeugt und aufrechterhalten:

> "The sense of place must be forged by the users; it cannot be inherent in the system itself."[152]

Dennoch kann der Systementwurf auf ein bestimmtes Platzverständnis ausgerichtet werden, um dieses zu unterstützen. Dabei genügt es nicht zu fragen, welche Aspekte der räumlichen Umgebung ausgenutzt werden, sondern muss stets gefragt werden, zu welchem Zweck sie genutzt werden[153].

In vielen Softwaresystemen wurde die Desktopmetapher von Einzelplatzsystemen zu einer Metapher von Schreibtischen, Büros, Korridoren und Städten ausgeweitet[154].

---

[146] Um ein anschauliches Beispiel zu geben: Die Turnhalle meiner Grundschule beherbergte z.B. regelmäßig eine Geflügelschau. Die ausgestellten Hühner, Enten und Gänse wären im normalen Schulbetrieb an diesem Ort "deplaziert" gewesen.
[147] [HarrisonDourish96], S.69
[148] [HarrisonDourish96], S.73
[149] [HarrisonDourish96], S.69
[150] [HarrisonDourish96], S.75
[151] [HarrisonDourish96], S.74
[152] [HarrisonDourish96], S.70
[153] [HarrisonDourish96], S.74
[154] [HarrisonDourish96], S.67

1.3.6 Ort und Platz

Abschließend möchte ich nun das Verständnis der zentralen Begriffe "Ort" und "Platz" definieren, so wie es dieser Arbeit zugrundeliegt.

Bei der Definition des Begriffes "Ort" berufe ich mich im Wesentlichen auf die Definition eines Ortes des locales frameworks in [Mansfield98][155] und auf die Merkmale der räumlichen Umgebung bei [HarrisonDourish96][156].

Begriff 13: Ort

Ein *Ort* ist eine räumliche Umgebung um einen "Brennpunkt", um den sich Dinge und Personen gruppieren.

Das räumliche Verständnis eines Ortes wird allgemein geteilt. Dinge und Personen werden an Orten unmittelbar wahrgenommen. Orte bieten die Möglichkeit zum Zugriff auf Dinge und zur Interaktion mit anderen Personen. Durch Entfernung oder durch explizite "Barrieren", wie z.B. Wände, können Orte voneinander abgegrenzt sein.

Beispiel für einen Ort ist eine Arbeitsumgebung.

Mit dem Begriff "Ort" verbindet sich eine gemeinsames Verständnis von Möglichkeiten und Strukturen. Dem entspricht das Verständnis der Arbeitsumgebung nach WAM als der Ort, an dem die Arbeitsmittel und -gegenstände für die Arbeit bereitliegen[157]. Der "Brennpunkt" einer Arbeitsumgebung ist dabei der Arbeitsplatz, an dem die eigentliche Arbeit stattfindet.

Plätze gestalten eine konkrete Arbeitsumgebung aus.

Begriff 14: Platz

Ein *Platz* ist ein Ort mit besonderer fachlicher Bedeutung – gemäß dem Sprachgebrauch "etwas gehört an seinen Platz", "ist deplaziert" usw.

Plätze sind begrenzt. Sie können Dinge, Personen und wiederum Plätze enthalten – sind also Behälter. Sie können sich nicht teilweise überschneiden.

An einen Platz sind stets Konventionen geknüpft. Diese beziehen sich z.B. auf dessen Inhalt, den Umgang damit, das erwartete Verhalten von Personen an diesem Platz, den Zugriff auf den Platz usw. Insbesondere kann ein Platz öffentlich oder privat sein.

Durch die Verknüpfung von Konventionen und räumlichen Eigenschaften ermöglichen Plätze die Orientierung in einer kooperativ genutzten Umgebung.

Typische Arten von Plätzen sind Arbeitsplätze[158], Gebäude, Räume[159], Möbel und Stapel.

[155] Kap.1.3.1, S.29
[156] Kap.1.3.5, S.39
[157] [WAM98], S.183
[158] im wörtlichen Sinn, z.B. ein Schreibtisch im Großraumbüro – nicht im übertragenen Sinne von Job / Position
[159] im Sinne von Zimmer

Die mit Plätzen verknüpften Konventionen grenzen Plätze klar voneinander und von anderen Orten ab[160]. In diesem Sinne sind Plätze begrenzt. Zudem sind sie immer in einen Kontext eingebettet, nämlich dem, von dem sie abgegrenzt werden. Die Grenzen eines Platzes sind i.d.R. an räumliche Merkmale geknüpft (z.B. Wände).

Da eine virtuelle Arbeitsumgebung von Menschen gestaltet wird, sind ihre räumlichen Bestandteile nicht bloße virtuelle Orte, sondern virtuelle Plätze. Die fachliche Bedeutung der Plätze fließt in die Gestaltung mit ein und wird zu einem Teil explizit modelliert.
In Softwaresystemen, die nach WAM in einem gemeinsamen Kommunikations- und Lernprozess zwischen Entwicklern und Benutzern entwickelt werden, haben die explizit modellierten Konventionen weiterhin ihren Ursprung in der Kultur des Anwendungsbereichs. Solche Konventionen in einem WAM-System explizit zu modellieren, ist gewissermaßen ein Weg, sie anzuwenden.
Nach Harrison und Dourish kann dem System nicht inhärent sein, wie Orte als Platz verstanden werden[161]. Dies ist insofern wahr, als sich das Verständnis der Benutzer ändern kann. Im übrigen kann dieses Verständnis nach meiner Auffassung sehr wohl in virtuellen Plätzen zum Ausdruck kommen.

[160] So existiert z.B. bei Marktplätzen eine klare räumliche Grenze. Innerhalb dieser Grenze findet der übliche Marktbetrieb statt und gilt die Marktordnung.
[161] Kap.1.3.5, S.40

1.4 Das JWAM-Framework

In diesem Unterkapitel möchte ich auf das JWAM-Framework eingehen.
Das JWAM-Framework (oder kurz JWAM für Java-Framework und WAM) ist eine Grundlage und Anleitung für die technische Realisierung interaktiver Anwendungssoftware nach dem WAM-Ansatz. Es stellt die Teile eines Systems bereit, die generisch jenseits eines konkreten Anwendungsbereichs liegen, und weist die Stellen aus, an denen eine anwendungsspezifische Konkretisierung anknüpfen kann[162].
Das JWAM-Framework bildet damit auch die Grundlage für den konstruktiven Teil dieser Arbeit, in dem eine Komponente für dieses Framework entwickelt wird

Das JWAM-Framework wird seit Mitte 1997 am Arbeitsbereich Softwaretechnik der Universität Hamburg von Studenten und wissenschaftlichen Mitarbeitern kontinuierlich (weiter-)entwickelt. Das Framework diente ursprünglich als Grundlage für Studien- und Diplomarbeiten, wurde bald aber auch in verschiedenen Lehrveranstaltungen verwendet. Mittlerweile wird das Framework auch in Kooperationsprojekten mit der Industrie eingesetzt. Seit Anfang 2000 wird es von der Apcon WPS GmbH mit Sitz in Hamburg in professionellem, forschungsnahem Rahmen weiterentwickelt[163].

Das JWAM-Framework ist vollständig in Java geschrieben. Dabei wird ein sehr konsistenter Coding Style Guide verwendet, der u.a. vorsieht, dass der funktionale Zusammenhang von Klassen bereits am Namen identifiziert werden kann – so beginnen z.B. die Bezeichner von Fachwertklassen mit "dv..." für (Domain Value) und die Werkzeugklassen mit "tool...".
In der ca. Mitte 2000 erschienenen Version 1.5 enthält das JWAM-Framework 72 Packages und 1159 Klassen bzw. Interfaces. JWAM 1.5 kann damit zu den mittelgroßen Softwaresystemen gezählt werden[164].

Das Unterkapitel gliedert sich in drei Teile. Bevor ich die Architektur des JWAM-Frameworks im Speziellen vorstelle, möchte ich in den beiden folgenden Teilen zunächst einen Überblick über die Framework- und Komponenten-Thematik im Allgemeinen geben.

1.4.1 Frameworks

Anwendungen können auf der Basis eines Frameworks schneller entwickelt werden. Werden sie auf der Basis desselben Frameworks entwickelt, so haben sie ähnliche Strukturen, sind einfacher zu warten und konsistenter. Objektorientierte Systeme erreichen mit Frameworks den höchsten Grad an Wiederverwendung[165].
Die Verwendung von Frameworks verspricht also eine erhöhte Produktivität, kürzere Entwicklungszeiten und Anwendungen von höherer Qualität. Dies macht verständlich, warum

[162] [JWAM99 1], S.91
[163] Weitere Informationen zu JWAM und das Framework selbst finden sich im Internet unter www.jwam.de.
[164] [SimonSteinbrückner00], S.7 und S.1. Schnittstellen wurden in dieser Untersuchung zu den Klassen gezählt.
[165] [Gamma97], S.31 und S.33

Frameworks zu einem wichtigen Bestandteil bei der Entwicklung großer objektorientierter Softwaresysteme geworden sind.

Der Hauptbeitrag eines Frameworks zu einer Anwendung ist die von ihm definierte *Architektur*[166].
Demzufolge kann ein Framework nicht für beliebige Anwendungen geeignet sein. Vielmehr werden nur bestimmte Klassen von Anwendungen durch die Architektur des Frameworks effizient unterstützt – man spricht hierbei oft von *Domänen*[167].
Die Architektur muss für alle unterstützten Domänen gleichermaßen funktionieren.

Frameworks ermöglichen die Wiederverwendung von Analysen, Entwurfsentscheidungen, Architekturen und Implementierungen[168].
Sie betonen somit die *Entwurfswiederverwendung* gegenüber der *Codewiederverwendung*. Ein Entwickler, der ein Framework benutzt, muss so weniger Entwurfsentscheidungen treffen und kann sich mehr auf die spezifischen Details der Anwendung konzentrieren – auf der anderen Seite verliert er dadurch aber einige kreative Freiheit[169].

Im Kontrast zu Frameworks können *Klassenbibliotheken*[170] gesehen werden. Eine typische Klassenbibliothek ist z.B. eine Behälter-Bibliothek.
Frameworks und Klassenbibliotheken bündeln beide eine Menge von Klassen. Letztere betonen jedoch die *Codewiederverwendung* und erzwingen keine bestimmte Anwendungsarchitektur: Wird eine Klassenbibliothek verwendet, so ruft die Anwendung den Code der Klassenbibliothek auf. Ihre Klassen lassen sich weitgehend (d.h. von Hilfklassen abgesehen) unabhängig voneinander verwenden.
Wird dagegen ein Framework verwendet, so ruft dieses den anwendungsspezifischen Code auf. Das Framework steuert quasi die Anwendung, indem sie das Zusammenspiel der Objekte zur Laufzeit, den Kontrollfluss festlegt[171].

Ein Großteil des Anwendungsentwurfs und -codes entstammt üblicherweise dem verwendeten Framework bzw. ist von ihm beeinflusst. Es besteht also eine große Abhängigkeit zwischen einem Framework und den darauf basierenden Anwendungen. Aufgrund dieser starken Abhängigkeit müssen sich Anwendungen mit einem Framework mitentwickeln, wenn dieses geändert wird[172].
Grundsätzlich wirken drei Größen auf die Anwendungs- und Frameworkentwicklung ein: der Anwendungsbereich, die verwendete Technik und die Handhabung und Präsentation. In diesen Dimensionen sollte ein Framework möglichst einfach und folgenlos verändert werden können[173].

---

[166] [Gamma97], S.32
[167] [JWAM99 1], S.92
[168] [Bäumer98], S.93
[169] [Gamma97], S.31
[170] oder auch *Toolkit* bei [Gamma97], S.30
[171] [Gamma97], S.31 und [Bäumer98], S.96
[172] [Gamma97], S.32f.
[173] [JWAM99 1], S.91

Da substantielle Änderungen an der Architektur von Nachteil sind, muss ein Framework also von vornherein möglichst flexibel und erweiterbar entworfen werden. *Lose Kopplung* ist dabei sehr wichtig. *Entwurfsmuster* erhöhen die Entwurfs- und Codewiederverwendbarkeit von Frameworks. Reife Frameworks enthalten üblicherweise viele Entwurfsmuster[174].

Ein Framework kann durch anwendungsspezifische Unterklassen spezialisiert werden. Oft enthält ein Framework auch konkrete Unterklassen, die sofort verwendet werden können[175].
Wiederverwendung durch Objektkomposition wird oft als *Black-Box-Wiederverwendung* bezeichnet. Die internen Details der verwendeten Objekte sind nicht sichtbar – sie erscheinen nur als "Black-Boxes". Wiederverwendung durch Unterklassenbildung nennt man im Gegensatz dazu oft *White-Box-Wiederverwendung* – dabei bezieht sich der Begriff "White-Box" auf die für die Unterklassen sichtbaren internen Gegebenheiten der Oberklassen[176].

Black-Box und White-Box sind keine statischen Eigenschaften eines Frameworks. Vielmehr wird ein Framework bei der Softwareentwicklung black-box- oder white-box-artig verwendet. Gleichwohl spricht man von einem *Black-Box-* bzw. *White-Box-Framework*, wenn es ursprünglich jeweils auf eine der beiden Verwendungsarten ausgerichtet ist[177].
Black-Box-Rahmenwerke entstehen oft aus White-Box-Rahmenwerken:

> "In fact, as the design of a system becomes better understood, black-box relationships should replace white-box ones. Black-Box relationships are an ideal towards which a system should evolve."[178]

Ein Framework legt die Stellen fest, an denen es erweitert und angepasst werden kann[179].
Die variablen Teile der Domäne einer Anwendung werden *Hot Spots* genannt, d.h. unterschiedliche Anwendungen für eine Domäne unterscheiden sich genau an diesen Hot Spots. Wird ein Großteil der Anwendung durch ein Framework realisiert, so liefert die Anwendung ihrerseits die verschiedenen Alternativen für die Hot Spots eines Frameworks[180].
Ein Hot Spot wird durch ein *Hot-Spot-Subsystem* implementiert. Dieses besteht aus der Schnittstelle, die den Hot Spot definiert, und konkreten Klassen, die diese Schnittstelle implementieren (ggf. enthält es auch noch weitere Klassen). Ein Hot-Spot-Subsystem wird black-box- oder white-box-artig in die vom Framework stammenden Teile der Anwendung eingebunden[181].

Zusammenfassend kann der Begriff "Framework" folgendermaßen definiert werden:

---

[174] [Gamma97], S.32.
[175] [Gamma97], S.31
[176] [Gamma97], S.22
[177] [Bäumer98], S.95f.
[178] R. E. Johnson, B. Foote. Designing Reusable Classes. The Journal of Object-Oriented Programming, 1(2). o.O. 1988. (S. 26) (zitiert in: [Bäumer98], S.95)
[179] [Bäumer98], S.93
[180] [Schmid97], S.48
[181] [Schmid97], S.48

Begriff 15: Framework

Ein Framework besteht aus eine Menge von Klassen und Schnittstellen und stellt einen wiederverwendbaren Entwurf für eine bestimmte Domäne dar.

Ein Framework bestimmt vor allem die Architektur einer Anwendung. Es definiert dessen Struktur im großen, seine Unterteilung in Klassen und Objekte, deren Zusammenarbeit und zentralen Zuständigkeiten sowie den Kontrollfluss[182].

Ein Framework kann durch Spezialisierung bzw. Komposition *white-box-* bzw. *black-box-artig* verwendet werden. Mit den sogenannten *Hot Spots* legt es seine variablen Teile fest.

1.4.2 Komponenten

Die folgende Definition des Begriffs der *Komponente* wurde zuerst 1996 auf der European Conference on Object-Oriented Programming (ECOOP) von Szyperski und Pfister formuliert:

Begriff 16: Komponente

"A software component is a unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to composition by third parties."[183]

Eine Komponente kapselt eine wiederverwendbare Einheit eines Softwaresystems. Über eine Schnittstelle bietet sie einen Satz von Dienstleistungen an.

Komponenten sind über Schnittstellen verbunden[184].
Bzgl. der Verwendung von Schnittstellen durch eine einzelne Komponente kann zwischen *incoming* und *outgoing interfaces* unterschieden werden. Die Unterscheidung bezieht sich dabei auf die Richtung, in der Anfragen durch die Schnittstelle gestellt werden: Eine Komponente empfängt in der Rolle des Anbieters Anfragen über ein incoming interface und stellt als Klient über ein outgoing interface Anfragen an eine verknüpfte Komponente. Betrachtet man eine Schnittstelle isoliert, so ist sie weder incoming noch outgoing interface – dies ist sie nur bzgl. einer gegebenen Komponente[185].

Schnittstellenspezifikationen können als Verträge zwischen der Komponente, die eine Schnittstelle implementiert, und der, die sie benutzt, angesehen werden. Die zwei Seiten eines Vertrages werden auf der Ebene von Operationen üblicherweise über Vor- und Nachbedingungen spezifiziert[186].

---

[182] [Gamma97], S.31
[183] [Szyperski99], S.34
[184] [Szyperski99], S.40
[185] [Szyperski99], S.149
[186] [Szyperski99], S.42f. Bei der Verwendung von Callback-Operationen ist dabei besonders zu beachten, dass sich der Anbieter während des Callback-Aufrufs in einem konsistenten sondierbaren Zustand befinden muss. Andernfalls verlieren die Vor- und Nachbedingungen einer Operation für den Klienten ihre Verbindlichkeit. ([Szyperski99], S.49)

In [WAM98] wird diese Interpretation einer Benutzt-Beziehung als ein vertraglich formal geregeltes Verhältnis von Leistungsanbieter und Klient als *Vertragsmodell* bezeichnet[187].

Änderungen an der Implementation einer Komponente sind unkritisch, sofern der zur Komponenten-Schnittstelle gehörende Vertrag nicht gebrochen wird. Vorbedingungen können dabei abgeschwächt, Nachbedingungen verstärkt werden[188].
Die Versionierung von Schnittstellen ist dagegen problematisch. Ein Lösungsansatz ist, Änderungen an einmal veröffentlichten Schnittstellen strikt zu verbieten. Eine neue Version eines Interfaces entspricht dann einem neuen Interface – veraltete Versionen werden nicht mehr unterstützt[189].

Der innere Aufbau einer Komponente bleibt "hinter" der Schnittstelle verborgen. Üblicherweise wird eine Komponente durch eine Struktur kooperierender Klassen realisiert, die z.B. im Sinne des Fassaden-Entwurfsmusters[190] über eine ausgezeichnete Klasse verwendet werden. In diesem Sinne lassen sich Komponenten als Frameworks auffassen. Komponenten sind jedoch nicht mit Klassen gleichzusetzen, da Klassen wesentlich feingranularer sind[191].

Die Implementierung einer Operation der Komponenten-Schnittstelle befindet sich in der Klasse des Objekts, das angesprochen wird, wenn die Komponente eine entsprechende Anfrage erhält. Die Implementation wird erst in diesem Moment ermittelt und ausgeführt (*late* oder *dynamic binding*). Late binding ermöglicht, Komponenten zur Laufzeit dynamisch zu verknüpfen[192].

Das oberste Ziel der Komponenten-Technologie ist der unabhängige Einsatz und Zusammenbau von Komponenten. Die Konstruktion einzelner Komponenten und eine grundlegende Unterstützung bei der Verknüpfung von Komponenten genügen nicht, um dieses Ziel zu erreichen. Dies kann erst durch *Komponenten-Frameworks* gelingen[193].

Ein Komponenten-Framework bietet eine Implementation, die bestimmt, wie Komponenten, die in das Framework eingebunden wurden, interagieren dürfen. Komponenten-Frameworks können entweder einzeln eine "Insel" für bestimmte Komponenten bilden, oder sie können ihrerseits mit anderen Komponenten-Frameworks kooperieren. Es ist somit nur natürlich, auch Komponenten-Frameworks als Komponenten zu modellieren und Komponenten-Frameworks höherer Ordnung zu schaffen, die das Zusammenspiel der einzelnen Frameworks regeln[194].

[187] [WAM98], S.93
[188] [Szyperski99], S.44
[189] [Szyperski99], S.43
[190] [Gamma97], S.189
[191] [Bäumer98], S.30
[192] [Szyperski99], S.41f.
[193] [Szyperski99], S.280
[194] [Szyperski99], S.280

In diesem Sinne geht der Komponentenbegriff über den Begriff des Hot-Spot-Subsystems hinaus.

1.4.3 Die JWAM-Schichtenarchitektur und ihre Frameworks

Größere objektorientierte Anwendungen bestehen sinnvollerweise aus Schichten von Frameworks, die zusammenarbeiten[195].
Dies bietet sich umso mehr für Frameworks an, da ein Framework naturgemäß recht komplex sein muss[196].
Innerhalb der verschiedenen Bereiche eines Frameworks existieren Abhängigkeiten, die eine Umsetzung der Struktur als Schichtenarchitektur nahelegen[197].
So ist auch JWAM in einer Schichtenarchitektur aufgebaut. Die JWAM-Schichtenarchitektur stellt eine logische Anordnung dar. Sie soll dem Entwickler eine prinzipielle Orientierung über den Ort von Entwurfsentscheidungen geben[198].
Das Hauptziel, das in JWAM mit der Schichtenarchitektur verfolgt wird, ist, Änderungen am Anwendungsbereich von Änderungen in der verwendeten Technik so unabhängig wie möglich zu machen[199].

In JWAM können protokollbasierte und objektorientierte Schichten unterschieden werden[200].
In einer protokollbasierten Schichtenarchitektur stellt jede Schicht nach oben eine Schnittstelle zur Verfügung und kapselt dadurch die darunterliegende Implementierung. Die anwendungsfernste Schicht liegt unten, die anwendungsnäheren Schichten weiter oben. Ziel dieser Anordnung ist es, dass höhere Schichten ausschließlich die Schnittstelle ihrer nächst niederen verwenden, und dass darunterliegende Schichten unsichtbar sind. Änderungen sollen dadurch möglichst lokale Auswirkungen haben[201].

Auch in einer objektorientierten Schichtenarchitektur sind die höheren Schichten anwendungsnäher als die niederen Schichten. Die höheren Schichten konkretisieren und spezialisieren die Konzepte der niederen Schichten, d.h. sie erben von diesen. Auch benutzen die höheren – wie bei der protokollbasierten Schichtenarchitektur – die niederen Schichten. Objektorientierte Schichten kapseln jedoch nicht die darunterliegenden, denn es ist durchaus möglich, eine nicht unmittelbar darunterliegenden Schicht zu verwenden[202].

Die Komponenten objektorientierter Schichten lassen sich erweitern, ohne selbst dabei verändert zu werden (Offen-Geschlossen-Prinzip)[203]. Bzgl. der White-Box-Verwendung ist eine Komponente offen, bzgl. der Black-Box-Verwendung geschlossen.

---

[195] [Gamma97], S.33
[196] [JWAM99 1], S.92
[197] [Bäumer98], S.101
[198] [JWAM99 1], S.92f.
[199] [Apcon JWAM o.J], S.1
[200] [JWAM99 1], S.92
[201] [JWAM99 1], S.92 und [WAM98], S.707
[202] [JWAM99 1], S.92
[203] [Bäumer98], S.105

Vererbung, Polymorphie und late binding überbrücken den Widerspruch zwischen Offen- und Geschlossenheit von Komponenten[204].

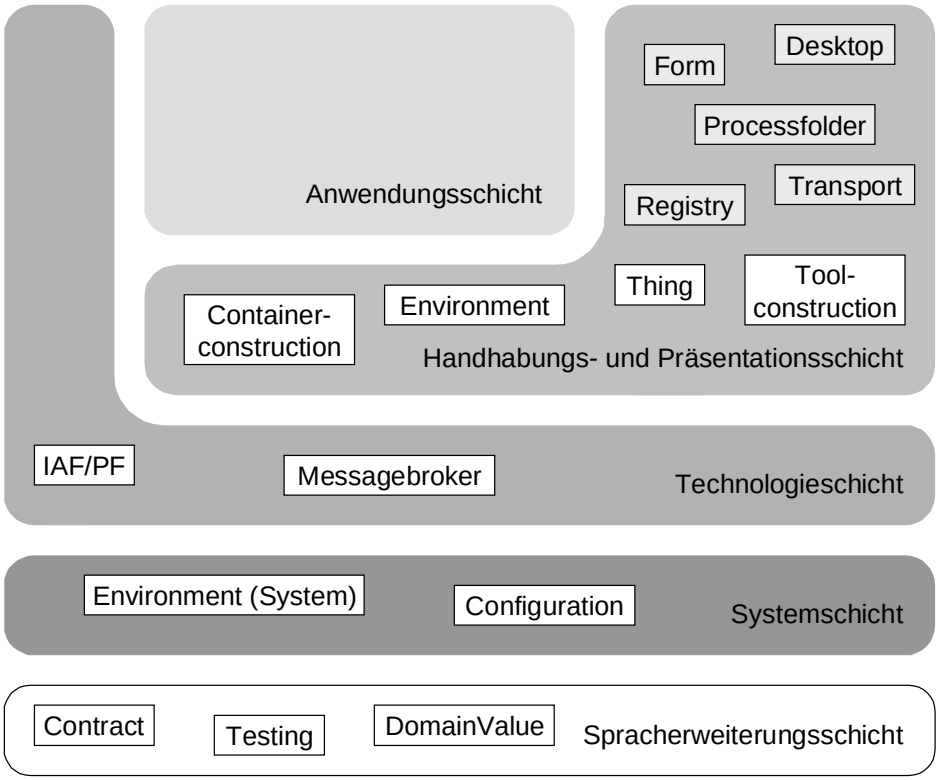


Abbildung 6: Die Schichten des JWAM-Frameworks[205]

Das JWAM-Framework besteht im einzelnen aus den folgenden Schichten (in Klammern jeweils die englische Bezeichnung, wie sie in der Implementation des Frameworks verwendet wird)[206]:

Spracherweiterungsschicht (language)

Die Spracherweiterungsschicht ergänzt die Sprache Java um einige zusätzliche Ausdrucksmöglichkeiten. Sie wird von allen anderen Schichten verwendet.

Die wichtigsten Frameworks dieser Schicht sind:

- *Contract*: Hier wird das Vertragsmodell[207] implementiert. Mit den Klassen dieses Black-Box-Frameworks werden die Vor- und Nachbedingungen aller Operationen sowie ggf. Klasseninvarianten überwacht.
- *Testing*: Dieses White-Box-Framework stellt eine Abstraktion des JUnit-Testing Frameworks für JWAM zur Verfügung. Von nahezu jeder Klasse des JWAM-Frameworks existiert eine korrespondierende Testklasse.

[204] [WAM98], S.719
[205] nach [Apcon JWAM o.J], S.1 und [JWAM99 1], S.95
[206] Quellen sind im folgenden – soweit nicht anders angegeben – die Artikel [JWAM99 1], [Apcon JWAM o.J], Kap.5.4.4 in [Bäumer98], S.111ff. sowie das JWAM-Framework selbst. Aufgrund der starken Ähnlichkeit der in den Quellen wiedergegebenen Informationen erscheint mir eine detaillierte Quellenangabe hier übertrieben.
[207] Kap.1.4.2, S.47

- *Domainvalue*: In diesem White-Box-Framework befinden sich Schnittstellen und abstrakte Basis-Klassen, um anwendungsspezifische Fachwerte[208] zu konstruieren. Zudem enthält es einen konkreten Fachwert für eindeutige Objekt-IDs. Fachwerte sind ein zentrales Konzept in JWAM und werden nicht nur in der Anwendungsschicht (s.u.) verwendet.

Von den Frameworks dieser Schicht – vor allem von den eben vorgestellten – wird in allen Schichten des JWAM-Frameworks sehr intensiv Gebrauch gemacht.

Systemschicht (system)

Die Systemschicht stellt als protokollbasierte Schicht Schnittstellen zur verwendeten Technik bereit. Diese umfasst z.B. das Betriebs- und Fenstersystem, Client/Server-Middleware oder relationale Datenbanken. Außerdem sind in ihr Garbage-Kollektoren, Mechanismen zur Serialisierung und Wiederherstellung von Objektgeflechten, Metaobjektprotokolle usw. angesiedelt.

Java und eng damit verbundene APIs stellen diese Schnittstellen und Mechanismen bereits weitgehend zur Verfügung – z.B. in Form der plattformunabhängigen Sprache Java selbst, der GUI-Klassenbibliotheken Swing und AWT, den Enterprise-APIs RMI (Remote Method Invocation) und JDBC (Java DataBase Connectivity), der Java-Garbagecollection, den in Java 1.1 neu eingeführten APIs zur Objektserialisierung und Reflexion[209] usw.

So beschränkt sich die Systembasisschicht des JWAM-Frameworks nur auf einige wenige Black-Box-Frameworks, wie z.B.:

- *Environment*: zur Kapselung des technischen Systems und für ein Standard-Exception-Handling und
- *Configuration*: zur Konfiguration der internationalen Sprachunterstützung.

Technologieschicht (technology)

Die Technologieschicht implementiert Modelle der verwendeten Technik. Sie verwendet die Systemschicht. Im Gegensatz dieser handelt es sich bei den Modellen der verwendeten Technik nicht um bloße Kapselungen, sondern um eigenständige, anwendungsnähere Konzepte.
Wenn die Technologieschicht auch noch anwendungsneutral verwendet werden kann, so ist sie doch schon von dem Leitbild und den Entwurfsmetaphern des WAM-Ansatzes geprägt.

Die Technologieschicht ist eine objektorientierte Schicht. Sie bietet z.B. die folgenden Subframeworks an:

- *Messagebroker*: Der Messagebroker ermöglicht, asynchrone Nachrichten über Prozessgrenzen hinaus zu verschicken. Die Klassen und Schnittstellen dieses

[208] Kap.1.1.3, S.22
[209] [Flanagan98], S.4, S.92, S.104ff.

White-Box-Frameworks gestatten eigene Implementationen des Messagebrokers. Ein Messagebroker auf der Basis von RMI wird mitgeliefert.

- *IAFPF*: Dieses Framework ist eine leistungsfähige Abstraktion des verwendeten GUI-Systems (bisher AWT oder Swing). GUI-Widgets werden in sogenannten *Präsentationsformen* (PF, Presentation Form) gekapselt, die ihrerseits von sogenannten *Interaktionsformen* (IAF, InterAction Form) verwendet und verändert werden können. Bei der Migration zu einem neuen GUI-System bleibt der Änderungsaufwand so auf die PFs beschränkt – Werkzeuge müssen nicht verändert werden, da sie ausschließlich IAFs verwenden.

Handhabungs- und Präsentationsschicht (handling)

Die Handhabungs- und Präsentationsschicht implementiert in zum Teil abstrakten Klassen die wiederverwendbaren, generischen Anteile der Entwurfsmuster der WAM-Metaphern. Damit befreit sie Entwickler von den lästigen Teilen bei der Konstruktion interaktiver Systeme. Sie ist die zentrale Schicht, um interaktive Systeme zu konstruieren. Gemessen an den enthaltenen Frameworks ist sie zudem die mit Abstand umfangreichste Schicht.

Die Handhabungs- und Präsentationsschicht verwendet als objektorientierte Schicht die Technologieschicht und die Systemschicht. Sie besteht aus den folgenden Frameworks:

- *Toolconstruction*: Dieses Framework enthält abstrakte Klassen und Schnittstellen für die Werkzeugkonstruktion. Dazu gehört etwa die Implementierung des Kontrollflusses innerhalb eines Werkzeugs oder ein ausimplementiertes Konzept zur Erzeugung von Werkzeugen und Sub-Werkzeugen.
- *Thing*: Dieses Framework dient der Konstruktion anwendungsspezifischer Gegenstände (Things). Die in diesem Framework enthaltene Thing-Schnittstelle ist von zentraler Bedeutung. Sie wird von allen Werkzeugen, Automaten, Materialien und Behältern implementiert.
  Die Thing-Schnittstelle sieht z.B. Operationen vor, um Dinge zu vergleichen, zu kopieren oder zu klonen sowie um die ID, den Namen und die Beschreibung eines Dinges – den sogenannten dvThingDescription-Fachwert – zu ermitteln.
- *Containerconstruction*: Dieses Framework bietet Schnittstellen und Klassen, um anwendungsspezifische Behälter (Container) zu konstruieren, wie z.B. Mappen, Ordner, Arbeitsflächen usw.
- *Environment*: Dieses Framework enthält eine vollständige Implementation der Arbeitsumgebung für die Werkzeuge, Automaten, Materialien und Behälter.

All diese Frameworks gehören zum Kern des JWAM-Frameworks. Sie werden generell bei der Anwendungsentwicklung auf der Basis von JWAM benötigt. Darüberhinaus bietet die Handhabungs- und Präsentationsschicht noch weitere

Frameworks an, die den Kern ergänzen und eher optionalen Charakter haben (in der Abb.6 werden sie daher etwas dunkler dargestellt). Dies sind z.B.:

- *Desktop*: eine fertig verwendbare Komponente nach der Schreibtisch-Metapher zur Visualisierung der Arbeitsumgebung (s.o) im Sinne eines virtuellen Einzelarbeitsplatzes,
- *Registry*: eine Registratur als anwendungsfachlich motiviertes Persistenzmedium, zum Registrieren und Aufbewahren von Materialien,
- *Transport*: ein Postversandsystem als Transportmedium, um Materialien asynchron zwischen Umgebungen auszutauschen,
- *Processfolder*: Vorgangsmappen mit den entsprechenden Werkzeugen und einem Vorgangsmonitor, um anwendungsspezifische Vorgänge zu unterstützen, und
- *Form*: um Formulare einfach zu konstruieren und mithilfe von mitgelieferten Standardwerkzeuge bearbeiten zu können.

Anwendungsschicht

Die Anwendungsschicht ist schließlich die Schicht, in der die Anwendung auf der Basis der anderen Schichten des Frameworks realisiert wird. Diese Schicht findet sich demzufolge nicht im JWAM-Frameworks selbst, wird aber im Zusammenhang mit den übrigen JWAM-Schichten vorgesehen. Insbesondere die Handhabungs- und Präsentationsschicht und die Technologieschicht geben der Anwendungsschicht einen Rahmen, in den sie eingebettet werden kann (dies soll in der Abb.6 durch die U-förmige Darstellung der beiden Schichten symbolisiert werden).

Die Aufteilung des JWAM-Frameworks in Schichten spiegelt sich in der Package-Struktur des Java-Codes wider (siehe Abb.7 auf der nächsten Seite). In Java ist jede Klasse oder Schnittstelle Teil eines Packages, das wiederum in einem Package geschachtelt sein kann. Die oben erwähnte Thing-Schnittstelle befindet sich z.B. im Package de.jwam.handling.thing.

Üblicherweise wird für das "Wurzelpackage" der Internet-Domain-Name der Organisation verwendet, die das Package entwickelt hat. Dies geschieht mit dem Ziel, global eindeutige Package-Bezeichner zu erhalten[210].

[210] [Flanagan98], S.17ff.

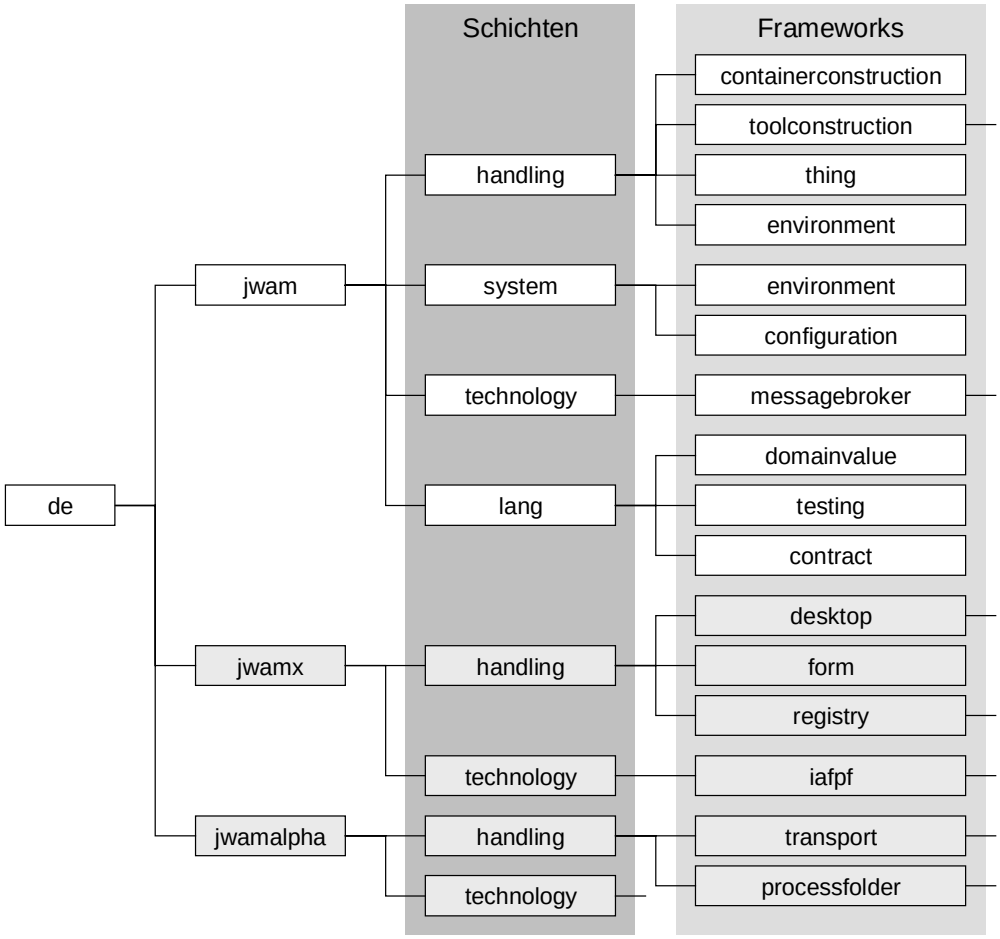


Abbildung 7: Package-Struktur des JWAM-Frameworks (Ausschnitt)[211]

In der Abbildung ist erkennbar, dass eine weitere Aufteilung die Aufteilung in Schichten überlagert. Zusätzlich wird noch zwischen dem Kern des JWAM-Frameworks (de.jwam), den bewährten Erweiterungen (de.jwamx) und den neuen Erweiterungen (de.jwamalpha) unterschieden (in der obigen Abbildung werden diese Erweiterungen – analog zur Abb.6 – etwas dunkler dargestellt). Der Kern enthält die Frameworks, die üblicherweise bei der Anwendungsentwicklung mit JWAM benötigt werden. Er kann unabhängig von den übrigen beiden Teilen verwendet werden. Die Erweiterungen basieren dagegen auf dem Kern und können ohne ihn nicht kompiliert werden.

[211] Die gezeigte Package-Struktur entspricht der im September 2000 erschienenen Version 1.5.0 des JWAM-Frameworks. Im konstruktiven Teil verwende ich die Version 1.5alpha, in der sich das Desktop-Framework noch im Package de.jwamalpha.handling befand.

1.5 COJAC – JWAM-Komponenten zur Kooperationsunterstützung

Seit März 1999 existiert das COJAC-Projekt am Arbeitsbereich Softwaretechnik des Fachbereichs Informatik der Universität Hamburg. Ziel des Projektes ist, Komponenten zur Kooperationsunterstützung für das JWAM-Framework zu entwickeln. Die folgenden Komponenten sind von den Mitgliedern des COJAC-Projekts im Rahmen von Diplomarbeiten ausgearbeitet und realisiert worden:

- das Postversandsystem von Mirko Freund[212],
- Vorgangsmappen und Vorgangsmonitor von Holger Breitling[213] und
- die Raumkomponente, die in dieser Arbeit behandelt wird.

Betreut wurde das COJAC-Projekt von Dr. Ingrid Wetzel.
Neben den COJAC-Komponenten ist die Registratur als JWAM-Komponente zur Kooperationsunterstützung zu nennen, die seit Mai 1999 im Framework existiert.

Das Postversandsystem besteht aus Postkörben, die sich in verschiedenen Arbeitsumgebungen befinden. Ein Zustelldienst und eine Postdienststelle, die sich an zentraler Stelle befindet und in der die zu versendenden Dinge in einer Postfachwand zwischengelagert werden, realisieren den Versand. Das Postversandsystem unterstützt so die explizite Kooperation durch Materialaustausch. Bei der expliziten Kooperation ist der Kooperationspartner explizit bekannt – hier z.B. in Form des Adressaten.

Auch die Registratur dient dem Materialaustausch. Sie unterstützt jedoch die implizite Kooperation, da die Kooperationspartner beim Materialaustausch nicht explizit bekannt sind.
Die Registratur realisiert ein fachlich motiviertes Persistenzkonzept. Über einen Registrar, der sich in den Arbeitsumgebungen der Kooperationspartner befindet, können Dokumente an der Registratur registriert, in ihr abgelegt und aus ihr entnommen werden. Dabei verwaltet die Registratur, welcher Benutzer ein bestimmtes Dokument entnommen hat.

Vorgangsmappen vergegenständlichen kooperative Arbeitsprozesse. Sie enthalten zu bearbeitende Dinge und einen Laufzettel. Der Laufzettel enthält eine Folge der typischen Aufgaben des Arbeitsprozesses. Diese können von den bearbeitenden Personen in ihrer Reihenfolge verschoben, durch weitere Aufgaben ergänzt oder ausgestrichen werden. Zudem werden erledigte Aufgaben abgehakt.
Vorgangsmappen werden z.B. über ein Postversandsystem oder eine gemeinsame Registratur zwischen den Benutzern ausgetauscht. Der Vorgangsmonitor ermöglicht dabei anderen Personen, Vorgangsmappen zu verfolgen und ihren letzten offiziellen Bearbeitungsstand zu ermitteln.

Die Raumkomponente unterstützt verschiedene Arbeits- und Kooperationsstile. Der Entwurf soll im folgenden erst genauer beschrieben werden, daher gebe ich hier nur eine kurze Übersicht über die unterstützten Arbeits- und Kooperationsstile:

---

[212] [Freund00]
[213] [Breitling00]

Wie das Postversandsystem und die Registratur kann die Raumkomponente zum Materialaustausch verwendet werden. Dabei unterstützt sie durch gemeinsame Räume die implizite und durch personengebundene Räume die explizite Kooperation.

Dadurch, dass der Zustand der Räume für alle Benutzer unmittelbar wahrnehmbar ist, wird die synchrone Kooperation unterstützt. Durch die persistente Natur von Räumen wird zudem die asynchrone Kooperation ermöglicht.

In Räumen ist sowohl die individuelle Arbeit als auch kooperative Arbeit möglich – je nachdem wieviele Benutzer in einem Raum arbeiten. Dies kann z.B. durch abschließbare Räume oder Konventionen geregelt werden. Zudem steht den Räumen der Einzelarbeitsplatz als spezieller Platz für die individuelle Arbeit gegenüber.

Durch die Möglichkeit zur expliziten Kommunikation, z.B. mündlich oder über eine eingebettete Chat- oder Audio-Verbindung, wird informale Kooperation unterstützt

Als Vorbild für die hier realisierte Raumkomponente diente innerhalb des WAM-Kontextes die WiGWAM-Raummetapher, die im Rahmen der Diplomarbeit von Stefan Roock und Henning Wolf entstanden ist[214]. Viele Teile meiner Arbeit sind unmittelbar von der WiGWAM-Metapher beeinflusst, und zu einem gewissen Teil sehe ich meine Arbeit als Fortführung der Arbeit von Roock und Wolf. Die Bezüge werden in der folgenden Arbeit im Einzelnen hergestellt.

Auch die Arbeit von Martin Lippert, der den JWAM-Desktop entwickelt hat, steht in engem Zusammenhang zu dieser Arbeit. Der JWAM-Desktop ist eine wichtige Grundlage für die Raumkomponente – in konzeptioneller wie softwaretechnischer Sicht. Beide Komponenten vergegenständlichen zusammen die individuelle Arbeitsumgebung eines Benutzers in einem kooperationsunterstützenden System. Auch hier werde ich die Bezüge in der folgenden Arbeit im Einzelnen herstellen.

[214] [RoockWolf98]

2. Fachliche Motivation für eine Raumkomponente

Im vorangegangenen Kapitel habe ich den Kontext beschrieben, in dem ich die zu entwickelnde Raumkomponente eingebettet sehe. Dabei ist an den drei vorgestellten Groupware-Systemen bereits erkennbar geworden, wie ein räumliches Konzept in einem Softwaresystem umgesetzt werden kann[215].
Die fachliche Motivation möchte ich nun in diesem Kapitel mit konkreten Anwendungsfällen illustrieren. Sie sollen plausible und typische Beispiele dafür geben, welche unterstützende Rolle der physische Raum in verschiedenen kooperativen Arbeitssituationen spielt. An ihnen möchte ich zeigen, wie eine Raumkomponente konzipiert werden kann, um dieselbe Unterstützung in der virtuellen Welt zu gewährleisten, die die Kooperationspartner der Anwendungsfälle in der physischen Welt vorfinden. Bei der Beschreibung des Entwurfs der Raumkomponente im nächsten Kapitel werde ich immer wieder auf diese Beispiele zurückkommen.

Die Anwendungsfälle stelle ich in Form von Szenarios vor, die jeweils eine bestimmte kooperativ zu lösende Aufgabe aus der Anwendungswelt in einem Prosatext beschreiben. Den Szenarios stelle ich jeweils ein Kooperationsbild[216] voran, in dem die beschriebene kooperative Arbeitssituation grafisch dargestellt ist. In dem Kooperationsbild werden allgemeinverständliche Piktogramme verwendet, deren genaue Bedeutung aus dem jeweiligen Szenario hervorgeht. Im übrigen bedeuten durchgezogene Pfeile, dass ein Material oder eine Information weitergeleitet wird – Nummern kennzeichnen dabei die Reihenfolge. Gestrichelte Pfeile symbolisieren, dass eine Ressource benutzt wird. Die Räumlichkeiten werden in Form eines Grundrisses visualisiert, der der tatsächlichen physischen Anordnung der Räume entspricht[217].

Die beiden Szenarios entstammen der Domäne Buchverlag. Da es mir hier nicht um eine umfassende Einführung in diese Domäne geht, erläutere ich die wenigen Fachbegriffe direkt im Text und verzichte auf ein ausführliches Glossar.

Die beiden Szenarios basieren auf Interviews, die ich im Rahmen dieser Arbeit mit einer ehemaligen Praktikantin in einem Buchverlag geführt habe. Sie sind alle mehrfach mit ihr rückgekoppelt worden und geben den gewählten Ausschnitt des damaligen Arbeitsalltags möglichst getreu wieder. Neben den Interviews dienten mir beispielhafte Exemplare der Arbeitsunterlagen (z.B. Mailings[218]) und der Praktikumsbericht der Praktikantin als primäre Informationsquellen.
Das dreimonatige Praktikum fand 1996 während des Germanistikstudiums der Praktikantin statt – zu jenem Zeitpunkt wurden nur wenige Arbeiten (z.B. das Drucken von

[215] Kap.1.3, S.27
[216] genauer: ein task picture, siehe Kap.1.1.1, S.14
[217] [RoockWolf98] bezeichnen diese Art von Kooperationsbildern, in denen Plätze bzw. in ihrem Sprachgebrauch Lokalitäten visualisiert werden, als *Lokalitätsbilder* ([RoockWolf98], S.77f.).
[218] Ein Beispiel für ein Mailing findet sich im Anhang. Zur Beschreibung eines Mailings siehe Szenario "Mailing Aktion", Kap.2.1.1, S.59.

Adressaufklebern) durch den Computer unterstützt. Die untersuchte kooperative Arbeit wurde durch bestehende Softwarelösungen meines Erachtens somit nicht merklich beeinflusst.

Die Domäne Buchverlag habe ich aus zwei Gründen gewählt. Zum einen beeindruckten mich die zahlreichen anschaulichen Beispiele für kooperative Büroarbeit, die sich bereits beim "ersten Kontakt" mit der Domäne finden liessen. Zum anderen hatte ich das Glück, jemanden zu kennen, der meine Fragen zur Kooperation im Buchverlag ausführlich beantworten konnte. Auf diese Weise konnte ich gezielt nach geeigneten Beispiele suchen, in denen die grundlegenden Anforderungen an eine Raumkomponente deutlich werden. Die beiden Szenarios, die ich schließlich gefunden habe, sind überschaubar und geben dennoch einen umfassenden und realistischen Einblick.

Der im WAM-Ansatz geforderten Anwendungsorientierung[219] kann ich mit den exemplarisch ausgewählten Szenarien und Kooperationsbildern meines Erachtens insoweit gerecht werden, als die in dieser Diplomarbeit konstruierte Raumkomponente am Anfang eines evolutionären Entwicklungsprozesses steht. Ich hoffe, dass sie weitere Entwicklungszyklen durchlaufen wird, in denen sie an differenziertere Anforderungen angepasst und weiter ausgestaltet werden wird.
Um für die zukünftige Weiterentwicklung das Erkennen von anderen Anwendungsfällen und die Übertragung darauf zu erleichtern, leite ich aus den Anforderungen der vorgestellten Anwendungsfälle jeweils ein verallgemeinertes Problem ab. Zu jedem dieser Probleme stelle ich die Konzepte vor, die im Anwendungsfall zur Lösung des Problems beitragen und beim Entwurf einer Raumkomponente somit berücksichtigt werden sollen.

2.1 Zwei Szenarios aus der PR-Abteilung eines Buchverlags

Der hier betrachtete Buchverlag besteht aus sechs Abteilungen, die zusammen in einem fünfstöckigem Gebäude untergebracht sind. Es arbeiten ca. 18 Mitarbeiter in diesem Gebäude – die Vertriebsabteilung hat zudem noch Mitarbeiter im Außendienst. Neben der Vertriebsabteilung gibt es noch die Abteilungen Rechte und Lizenzen, das Lektorat, die Setzerei, die Werbung und schließlich die PR-Abteilung (kurz PR), aus der die betrachteten Szenarios stammen.

PR steht für "Public Relations", die volle deutsche Bezeichnung lautet "Abteilung für Presse- und Öffentlichkeitsarbeit". Das Ziel der PR ist, die Medien (Printmedien, TV, Internet usw.) dazu zu veranlassen, Artikel zu einem Produkt des Verlags (Buchreihe, Buch, Autor usw.) zu veröffentlichen. Im Gegensatz zur Werbung spricht die PR die Öffentlichkeit nicht direkt an (z.B. über Anzeigen), sondern bedient sich dabei der Medien – Werbung findet zudem aus Kostengründen oft gar nicht oder nicht im nötigen Umfang statt[220]. Darüber hinaus ist die PR die Schnittstelle des Buchverlags nach außen und dient der Öffentlichkeit, vor allem aber den Medien als Anlaufstelle für Informationen über den Verlag.

[219] [WAM98], S.1095
[220] [Röhring92], S.150

Ca. 2100 Verlage[221] publizierten 1999 mehr als 80.799 Neuerscheinungen (Erst- und Neuauflagen)[222]. Es ist für die PR eines Verlags also nicht leicht, Aufmerksamkeit für ein neues Buch zu erregen. Entsprechend aufwendig betreut die PR die einzelnen Journalisten: Sie verschickt Mailings zu neu erschienenen Büchern, verteilt kostenlose Musterexemplare, schreibt und verschickt druckreife Artikel zu den Büchern (sogenannte Waschzettel[223]), veranstaltet Medienevents und vermittelt Interviews mit den Autoren.

Aber auch für die übrigen Abteilungen des Buchverlags erfüllt die PR eine wichtige Funktion, indem sie die Belegexemplare der Medien (Kopien von erschienenen Artikeln, Rundfunkmanuskripte, Screenshots von HTML-Seiten usw.) sammelt und regelmäßig zu einem sogenannten Pressespiegel zusammenstellt, der im Umlauf durch die Abteilungen des Buchverlags geht.[224]

Zur PR zählen vier Mitarbeiterinnen:
- eine Abteilungsleiterin,
- zwei weitere Mitarbeiterinnen und
- eine Praktikantin.

Im Buchverlagsgebäude teilt sich die PR eine Etage mit der Werbung, wobei die Räumlichkeiten der PR drei Räume umfassen:
- das Arbeitszimmer der Abteilungsleiterin,
- das Sekretariat und
- den Waschzettelraum – einen Raum, in dem alle Waschzettel archiviert werden.

Letzterer wird nicht als Arbeitsraum genutzt. Das Sekretariat ist Arbeitsraum der beiden Mitarbeiterinnen und der Praktikantin (zusammenfassend nenne ich sie daher die Mitarbeiterinnen des Sekretariats).

Die Aufgaben sind unter den vier Mitarbeiterinnen relativ gleich verteilt (dies gilt auch für die Praktikantin, da sie bereits über einige Erfahrung mit den Aufgaben aus anderen Praktika verfügt). Alle Mitarbeiterin sind prinzipiell für die gleichen Aufgaben zuständig und können sie ähnlich qualifiziert erfüllen. Zu diesen Aufgaben zählen das Schreiben und Verschicken der Mailings (siehe Szenario "Mailing-Aktion"), der Telefondienst, das Sammeln der Anfragen von Journalisten nach Musterexemplaren (siehe Szenario "Anfragen annehmen"), das Schreiben der Waschzettel und das Verschicken derselben zusammen mit den Musterexemplaren (die sogenannte Bemusterung) und schließlich das Erstellen des Pressespiegels. Lediglich die Abteilungsleiterin hat – neben der Leitung der Abteilung – noch weitere Aufgaben, wie z.B. das Vermitteln von Interviews, die Organisation von Medienevents (z.B. Buchmesse) usw. Daher delegiert sie des öfteren einzelne Aufgaben (siehe Szenario "Mailing-Aktion") bzw. überlässt bestimmte Aufgaben, wie z.B. den

[221] laut Selbstdarstellung auf der Homepage des Börsenvereins des deutschen Buchhandels e.V. (siehe http://www.boersenverein.de/bverein.htm)
[222] Seit der Wiedervereinigung ist die Titelproduktion mit Schwankungen um 19% gestiegen ([Börsenverein00, S.57). Zur Zeit der Szenarios, 1996, läge die Titelproduktion also rein arithmetisch bei ca. 74.000 Neuerscheinungen – in jedem Fall eine beachtliche Größenordnung.
[223] Neben dem duckreifen Text enthält der Waschzettel auch die bibliographischen Angaben – also Autorenname, Haupt- und Untertitel, Format, Umfang, Ladenpreis usw. ([Röhring92], S.152).
[224] [Röhring92], S.155

Telefondienst, generell eher den Mitarbeiterinnen des Sekretariats. Alle Mitarbeiterinnen sind zudem oft aushäusig, um persönliche Kontakte zu pflegen und Informationen an entscheidende Personen zu bringen.[225]

2.1.1 Szenario "Mailing-Aktion"

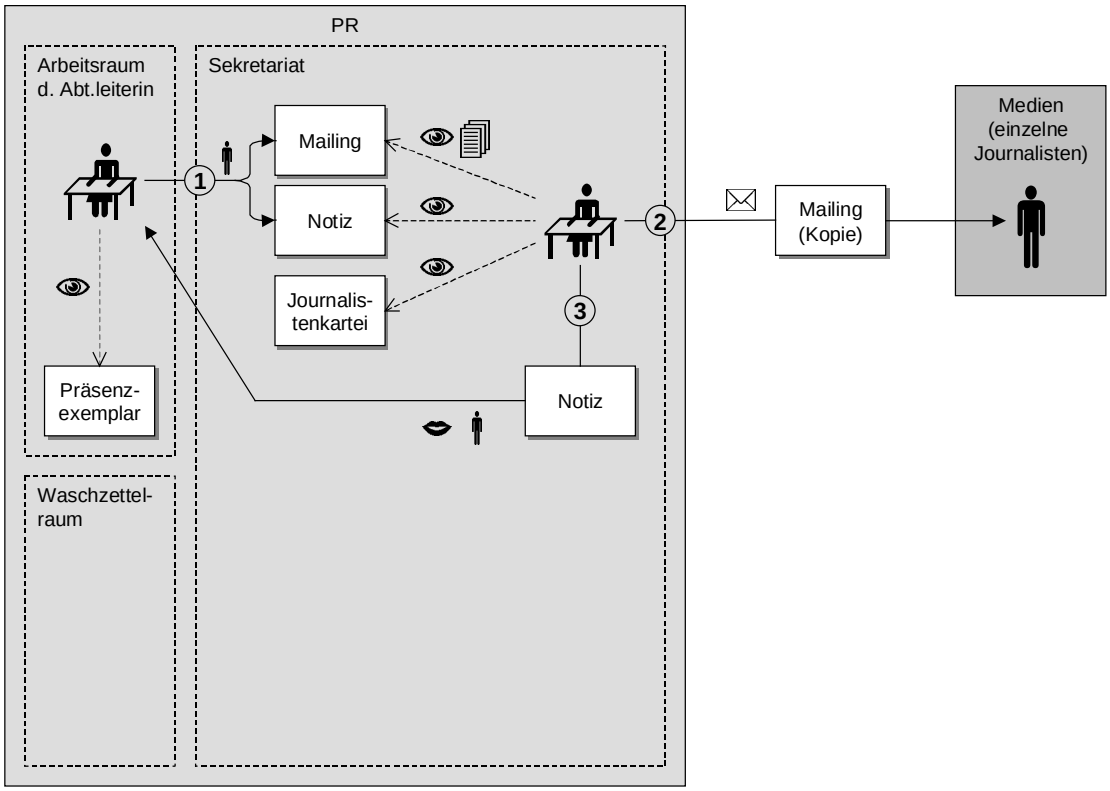


Abbildung 8: Kooperationsbild "Mailing-Aktion"

Die Abteilungsleiterin erhält aus der externen Druckerei ein Exemplar eines neuen Buchs, das sogenannte Präsenzexemplar. Sie liest die Klappentexte und einige Seiten, um einen Eindruck vom Inhalt zu gewinnen. Anschließend schreibt sie an ihrem Arbeitsplatz ein Mailing zu dem Buch. Das ist ein kurzes Anschreiben, in dem das Buch angekündigt wird, gefolgt von einer knappen Inhaltsangabe und einigen zusätzlichen Informationen, wie z.B. dem Umfang des Buches. Mailings werden jeweils an ca. 100-120 ausgewählte Journalisten verschickt, um sie über das neu erschienene Buch in Kenntnis zu setzen. Dies macht die Abteilungsleiterin in diesem Fall jedoch nicht selbst, sondern delegiert diese Aufgabe an eine beliebige Mitarbeiterin des Sekretariats. Dazu versieht sie das von ihr geschriebene Mailing mit einer Notiz, in der sie um die weitere Erledigung bittet, und legt beide Dinge ins Sekretariat.
Sobald eine der Mitarbeiterinnen Zeit findet, um nachzuschauen, was die Abteilungsleiterin vorbeigebracht hat, liest sie die Notiz und das Mailing. Daraufhin nimmt sie beide Dinge an ihren Arbeitsplatz und beginnt, den Verteiler für die Mailing-Aktion zu erstellen. Dazu durchsucht sie die Journalistenkartei nach solchen Journalisten, die sich für Bücher der Kategorie interessieren, in die das Buch aus dem Mailing fällt. Anschließend macht sie für

[225] [Röhring92], S.156

jeden dieser ausgewählten Journalisten eine Kopie des Mailings und verschickt diese Kopie per Post.
Nachdem die Mitarbeiterin des Sekretariats die Aufgabe erfüllt hat, informiert sie die Abteilungsleiterin darüber mündlich oder legt bei ihrer Abwesenheit eine Notiz in ihren Raum.

2.1.2 Szenario "Anfrage annehmen"

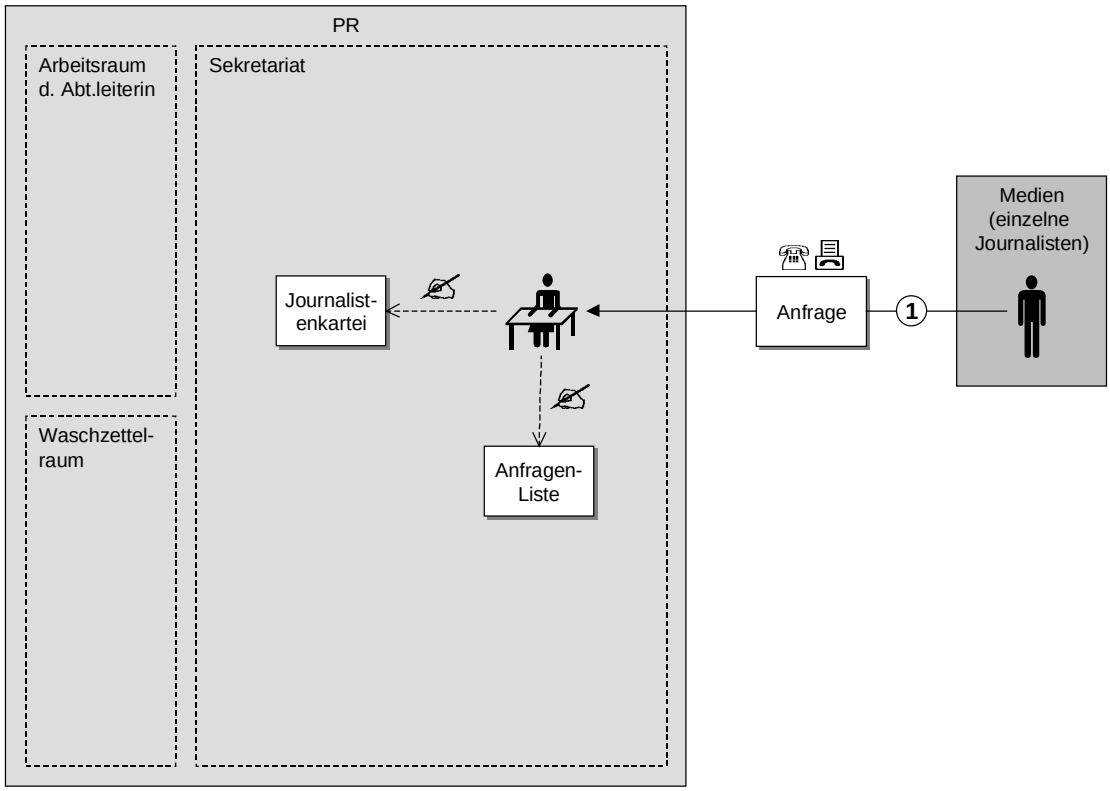


Abbildung 9: Kooperationsbild "Anfrage annehmen"

Ein Journalist fragt bei der PR telefonisch um ein kostenloses Musterexemplar eines Buches an (dies kann auch per Fax geschehen). Eine Mitarbeiterin des Sekretariats nimmt das Gespräch entgegen und notiert auf der Anfragenliste den Namen des Journalisten und das Buch, um das er angefragt hat. Die Anfragenliste wird täglich erstellt, von allen Mitarbeiterinnen, die Anfragen entgegennehmen, laufend ergänzt und am Ende des Arbeitstages von einer Mitarbeiterin abgearbeitet. Sie enthält dann ca. 30 Anfragen.
Ggf. notiert sich die Mitarbeiterin, die die Anfrage entgegennimmt, auch die Adresse des Journalisten, falls er nicht über die Mailing-Aktion von dem Buch erfahren hat und demnach in der Journalistenkartei noch nicht als Interessant für die Kategorie des Buches eingetragen worden ist. In diesem Fall aktualisiert sie anschließend die Journalistenkartei.

2.2 Anforderungen und Konzepte

Die vorgestellten Szenarien geben für verschiedene Arten der Kooperation Beispiele. Nicht alle davon möchte ich durch eine Raumkomponente unterstützen, da diese Arten besser durch andere, zum Teil schon realisierte JWAM-Komponenten unterstützt werden:

- explizite asynchrone Kooperation zwischen PR und Medien via Post- oder Faxversand – siehe JWAM-Postversandsystem[226]
- implizite asynchrone Kooperation über gemeinsame Archive (Journalistenkartei) – siehe JWAM-Registratur[227]

Beinahe selbstverständlich erscheint, dass die Medien, die diese Kooperationsarten unterstützen, und die übrigen Arbeitsmittel allen Beteiligten in einer gemeinsamen Umgebung (den Räumlichkeiten der PR) zur Verfügung stehen. Gerade in diesem Punkt greifen auf Einzelarbeitsplätze beschränkte Systeme aber zu kurz, da sie in ihrem Benutzungsmodell diese gemeinsame Umgebung nicht vergegenständlichen.
Kooperationsmedien, wie die eben genannten, und Ressourcen stehen in Einzelplatzsystemen ohne einen einbettenden Kontext zur Verfügung, in dem diese verstanden werden können.

Welche Anforderungen hat die gemeinsame Umgebung im Einzelnen zu erfüllen? Warum genügt es z.B. nicht, sich auf die Modellierung der in der gemeinsamen Umgebung befindlichen Ressourcen zu beschränken und "irgendeinen" Zugriffspunkt darauf anzubieten?
In den folgenden Unterkapiteln möchte ich anhand von "Schlüsselszenen" aus dem Szenario, die jeweils ein bestimmtes Problem bei der kooperativen Arbeit illustrieren, die Anforderungen an eine gemeinsame Umgebung herausstellen. Zu jeder Anforderung existiert im Szenario ein konkretes Konzept, das die kooperative Arbeit in der beschriebenen Form ermöglicht und daher später im Entwurf der Raumkomponente wieder aufgeriffen wird. Diese Konzepte und ihren Zweck fasse ich am Ende der folgenden Unterkapitel jeweils in einer kleinen Tabelle zusammen.

2.2.1 Koordination über Material und Platz

Im ersten Szenario, "Mailing-Aktion", legt die Abteilungsleiterin ein Mailing mit Notiz ins Sekretariat, um die Aufgabe, das Mailing zu verschicken, an eine beliebige Mitarbeiterin des Sekretariats zu delegieren. Wenn eine ihrer Mitarbeiterinnen das Mailing mit Notiz im Sekretariat findet, weiß sie, dass sie sich diese Dinge anschauen und etwas damit machen soll, da dies die geltende Konvention ist.
Die Rolle des besonderen Ortes wird deutlich, wenn man das Szenario geringfügig abwandelt. Lässt die Abteilungsleiterin Mailing und Notiz auf dem Schreibtisch in ihrem Arbeitszimmer liegen, so ist dies eine vollkommen andere Situation: eine Mitarbeiterin, die beide Dinge auf dem Schreibtisch der Abteilungsleiterin entdeckt, wird diese ignorieren, da es für sie nicht den Konventionen entspricht, unaufgefordert Dinge vom Arbeitsplatz der Abteilungsleiterin einzusehen oder zu entfernen.

---

[226] [Freund00]
[227] [JWAM 1]

Die Delegation der Aufgabe, das Mailing zu verschicken, ist nicht die einzige Szene, in der Kooperation über Material und Platz stattfindet. Auch die Mitarbeiterinnen des Sekretariats koordinieren sich in diesem Szenario untereinander über Material und Platz: Sobald sich eine der Mitarbeiterinnen das Material nimmt und auf ihren Arbeitsplatz holt, übernimmt sie damit auch die mit diesen Dingen verknüpfte Aufgabe. Sie muss sich nicht weiter mit ihren Kolleginnen abstimmen, da diese das Mailing mit der Notiz nicht mehr an dem Platz finden, an dem es von der Abteilungsleiterin abgelegt wurde.

An dieser Stelle sei ein kurzer Blick auf die derzeitige Situation der computergestützten Kooperation gestattet: Bisher wird fast nur die explizite Kooperation angemessen unterstützt (z.B. durch E-Mail / Postversand) – dies führt zu solchen Phänomenen, dass z.B. 45% aller Mails an den Zimmernachbarn verschickt werden[228]. Für den im Szenario geschilderten Fall stelle ich aber fest, dass implizite Kooperation über einen gemeinsamen Platz deutlich effizienter ist als explizite Kooperation: Stünde der Abteilungsleiterin aus dem Szenario z.B. nur der Postversand zum Materialaustausch zur Verfügung, so hätte sie an jede Mitarbeiterin eine Kopie des Mailings und der Notiz verschicken müssen, um die Aufgabe an eine der Mitarbeiterinnen zu delegieren. Die Mitarbeiterinnen hätten daraufhin ihrerseits den Koordinationsbedarf erkennen müssen (z.B. indem die Abteilungsleiterin in der Notiz darauf hinweist), um sich schließlich zu einigen, wer von ihnen die Aufgabe annimmt. Die Mitarbeiterinnen müssen sich abstimmen, welche der Kopien des Mailings und der Notiz zum "neuen" Original wird und bearbeitet werden soll. Tun sie dies nicht, besteht die Gefahr, dass die Aufgabe von mehreren Mitarbeiterinnen bearbeitet wird.
Kopien ziehen immer Koordinationsbedarf nach sich, wenn zu einem späteren Zeitpunkt eine davon zum Original gemacht werden soll. Für den im Szenario geschilderten Anwendungsfall werden unnötige Kopien durch die Kooperation über den gemeinsamen Platz vermieden.

Ein gemeinsamer Platz allein genügt jedoch noch nicht, um sich darüber bereits koordinieren zu können. Es müssen dazu verlässliche Bedingungen gelten, die es den Beteiligten ermöglichen, sich an diesem Platz zurechtzufinden, die (nicht von ihnen selbst ausgelösten) Geschehnisse zu verstehen und um sich vorzustellen zu können, wie sich der gemeinsame Platz für andere Personen darstellt. Dies ist in der physischen Welt aufgrund der Naturgesetze und unserer fundierten Erfahrung mit eben diesen möglich. Diese Naturgesetze gelten in der "virtuellen Welt" jedoch nicht ohne weiteres, ihr Effekt muss erst nachgebildet werden. Daher werde ich sie nun so weit sie relevant erscheinen genauer betrachten und bei der Konzeption der Raumkomponente berücksichtigen.

Für die Mitarbeiterinnen der PR ist selbstverständlich, dass sie die gemeinsame Arbeitsumgebung physisch auf die gleiche Weise wahrnehmen. Obwohl sich die Abteilungsleiterin und die Mitarbeiterinnen des Sekretariats aufgrund ihrer unterschiedlichen Rollen ein unterschiedliches Bild von ihrer gemeinsamen Arbeitsumgebung und ihren Handlungsmöglichkeiten darin machen, wissen alle Beteiligten, dass sie die physische Seite der Dinge, Personen und Räumlichkeiten auf dieselbe Weise wahrnehmen – diese physische Seite ist der kleinste gemeinsame Nenner bei der Wahrnehmung der gemeinsamen

[228] [Oberquelle98], S.101

Umgebung. Ein solcher gemeinsamer Nenner ermöglicht es der Abteilungsleiterin und den Mitarbeiterinnen des Sekretariats, sich über die gemeinsame Arbeitsumgebung zu verständigen. Sie können so z.B. Kooperationspartnern mitteilen, an welchem Platz sich etwas befindet, neue Konventionen im Umgang mit der gemeinsamen Umgebung abstimmen usw.

Für die Koordination über Material und Platz ist es notwendig, dass ein Kooperationspartner den Handlungsrahmen des anderen abschätzen kann. Dafür ist es notwendig, dass jedem Kooperationspartner klar ist, welche Bedingungen innerhalb der gemeinsamen Umgebung für die anderen Kooperationspartner gelten – dies sind auf alle Fälle die allgemein gültigen physischen Gesetze. So ist es nur natürlich, dass eine Mitarbeiterin des Sekretariats auf Mailing und Notiz an dem Ort zugreifen kann, an dem sie sie entdeckt hat. Die Abteilungsleiterin weiß dies und kann somit sicher sein, dass die Mitarbeiterin, die Mailing und Notiz an dem Platz findet, an dem sie sie hinterlegt hat, findet, sich beide Dinge nehmen und bearbeiten kann.

Ebenso natürlich ist, dass sich das Mailing zu einem bestimmten Zeitpunkt nur an genau einem Platz befinden kann – man spricht hierbei von der Einheit von Raum und Zeit[229].
Ohne die Einheit von Raum und Zeit wäre es den Mitarbeiterinnen des Sekretariats z.B. nicht möglich, sich über das Mailing und die Notiz zu koordinieren, indem sich eine von ihnen einfach beides nimmt. Bestünde rein theoretisch die Einheit von Raum und Zeit nicht, wäre es u.U. möglich, dass eine andere Mitarbeiterin zur selben Zeit an einem anderen Ort auf das Mailing zugreift und ebenfalls die damit verbundene Aufgabe, das Mailing zu verschicken, übernimmt. Beide Mitarbeiterinnen könnten die Konkurrenzsituation weder unmittelbar bemerken, noch könnten sie sich koordinieren, indem sich einfach eine von ihnen das Mailing nähme.

Erst durch die Einheit von Raum und Zeit ist es sinnvoll, von Plätzen und Orten statt von bloßen "Zugriffspunkten" zu sprechen, die das System anbietet[230]. Wäre es nicht eindeutig, an welchem Punkt im Raum sich eine Ressource zu einem bestimmten Zeitpunkt befindet, wäre es sinnlos, an diesen "Zugriffspunkt" Konventionen im Umgang mit der Ressource zu knüpfen, da zur selben Zeit an einem anderen "Zugriffspunkt" auf die Ressource möglicherweise widersprüchliche Konventionen gelten. Die Einheit von Raum und Zeit macht den "Zugriffspunkt" zu einer aussagekräftigen Eigenschaft einer Ressource und ermöglicht die räumliche Vorstellung von Plätzen, wie wir sie aus der physischen Welt kennen.

[229] "Raum" ist hier im Sinne der "gesamten räumlichen Umgebung" gemeint, nicht im Sinne von "Zimmer". [WAM98], S.878 spricht von der "Einheit von Ort und Zeit" – meint aber dasselbe.

[230] Zur Bedeutung des Begriffs "Zugriffspunkt" in diesem Absatz: In Win95 werden die Dateien, die sich in einem bestimmten System-Verzeichnis befinden, auf dem Desktop visualisiert. Das Verzeichnis und der Desktop sind beides "Zugriffspunkte". Eine Einheit von Raum und Zeit existiert nicht, da Dinge sich an beiden "Zugriffspunkten" gleichermaßen befinden. Insofern kann man das Verzeichnis und den Desktop nicht als Ort bezeichnen. Dennoch kann man auf die entsprechenden Dateien über diese beiden "Zugriffspunkte" zugreifen.

Konzept	Zweck
gemeinsame Umgebung	für den Materialaustausch
unterschiedliche Plätze in der gemeinsamen Umgebung	um die gemeinsame Umgebung und die Arbeit darin strukturieren zu können
alle Dinge an einem gemeinsamen Platz sind unmittelbar und für jeden an diesem Platz gleichermaßen sichtbar...	um sich über den gemeinsamen Platz verständigen zu können
... und zugreifbar	um den Handlungsrahmen eines Kooperationspartners einschätzen zu können
Einheit von Raum und Zeit	um eine räumliche Vorstellung von Plätzen zu ermöglichen

Tabelle 3: Konzepte zum Problem "Koordination über Material und Platz"

2.2.2 Gemeinsame Nutzung knapper Ressourcen

In den ersten beiden Szenarios, "Mailing-Aktion" und "Anfrage annehmen", benötigen zwei Mitarbeiterinnen des Sekretariats die Journalistenkartei für unterschiedliche Aufgaben.
Beide Mitarbeiterinnen kennen sich in ihrer Arbeitsumgebung aus und wissen, an welchem Platz sie die Journalistenkartei finden – üblicherweise befindet sie sich im Sekretariat. Da die Journalistenkartei von allen Mitarbeiterinnen für verschiedene Aufgaben benötigt wird und ihnen somit jederzeit bekannt sein muss, an welchem Platz sie auf die Journalistenkartei zugreifen können, entspricht es den Konventionen, sie nicht von diesem gemeinsamen Platz zu entfernen. Die Journalistenkartei gehört mit anderen Worten zum Inventar des Sekretariats. Die Topologie des Sekretariats ist stabil, soweit es die Arbeitsmittel betrifft, die zum Inventar des Sekretariats gehören.

Ich spreche hier von einer *stabilen* Topologie, da sich Konventionen ändern können, die es notwendig machen, das Inventar eines Raumes zu verändern. Die Szenarios geben dafür kein Beispiel, es ist aber leicht vorstellbar, dass die Journalistenkartei zukünftig in einem anderen Raum stehen soll, z.B. im Arbeitsraum der Abteilungsleiterin. Die Topologie darf also nicht starr sein (so wie z.B. die Nachbarschaftstopologie der Räume, die von den Mitarbeiterinnen nicht verändert werden kann) und auch nicht vollkommen variabel (so wie z.B. die Enthaltenseintopologie der nicht inventarisierten Dinge in den Räumen, die von den Mitarbeiterinnen jederzeit in andere Räume mitgenommen werden können). Die stabile Topologie spiegelt die geltenden Konventionen wider, welche Dinge sich an welchem Platz zu befinden haben. Sowohl die starre Topologie der Räume, als auch die stabile Topologie des Inventars dienen den Mitarbeiterinnen zur Orientierung in ihrer gemeinsamen Arbeitsumgebung.

Es kann vorkommen, dass zwei Mitarbeiterinnen zur gleichen Zeit mit der Journalistenkartei arbeiten möchten. In diesem Fall müssen sich die Mitarbeiterinnen koordinieren, da die

Journalistenkartei zu einem Zeitpunkt nur von einer Mitarbeiterin benutzt werden kann. Wenn eine Mitarbeiterin bemerkt, dass die Journalistenkartei gerade von einer Kollegin verwendet wird, wartet sie i.d.R., bis sie sieht, dass diese ihre Arbeit damit beendet hat. Die notwendige Koordination, um die Konkurrenzsituation zu bewältigen, liegt in der Verantwortung der beiden Kooperationspartnerinnen und wird durch Konventionen unterstützt.
Die Mitarbeiterinnen können die Handlungen der anderen unmittelbar erkennen. Auf diese Weise können Konkurrenzsituationen von den Beteiligten nicht nur bemerkt, sondern in gewissen Grenzen auch im Voraus abgesehen werden.
Außerdem erkennen die Mitarbeiterinnen, mit wem sie jeweils in einem Konflikt um die Journalistenkartei stehen. Kooperationspartner müssen sich in einer Konkurrenzsituation erkennen können, um in der Lage zu sein, miteinander explizit zu kommunizieren.
Sowohl die unmittelbare Wahrnehmung der Handlungen als auch der Handelnden gewährt den Kooperationspartnern, die Konkurrenzsituation differenzierter einschätzen zu können und nach eigenem Ermessen zu reagieren.

Die Möglichkeit, dass sich die Handelnden gegenseitig und ihre Handlungen wahrnehmen können, bezeichnet man als *Awareness*[231]. Diese Awareness ist in der physischen Welt selbstverständlich und darf bei der Computerunterstützung nicht verloren gehen.

Konzept		Zweck
gemeinsame Plätze		als Zugriffspunkte auf gemeinsame Ressource
starre Topologie		zur Orientierung in der gemeinsamen Arbeitsumgebung
stabile Topologie innerhalb einer starren Topologie (Inventar)		um den Zugriffspunkt auf gemeinsame Ressource innerhalb der gemeinsamen Arbeitsumgebung festlegen zu können
Awareness	in der gemeinsamen Umgebung sind die Handlungen der anderen unmittelbar erkennbar und...	um Konkurrenzsituationen abzusehen und differenzierter einschätzen zu können
	... können sich die Handelnden gegenseitig unmittelbar erkennen	um mit dem Kooperationspartner explizit kommunizieren zu können und um Konkurrenzsituationen differenzierter einschätzen zu können

Tabelle 4: Konzepte zum Problem "gemeinsame Nutzung knapper Ressourcen"

2.2.3 Handlungssicherheit

Im ersten Szenario schreibt die Abteilungsleiterin an ihrem Arbeitsplatz das Mailing. Diese Arbeit erfordert keine Kooperation und wird von der Abteilungsleiterin nicht in der öffentlichen gemeinsamen Umgebung, sondern an ihrem privaten Arbeitsplatz verrichtet. Erst

[231] [RoockWolf98], S.28

das Ergebnis veröffentlicht sie, indem sie das Mailing zusammen mit der Notiz in das Sekretariat legt.
Roock und Wolf bemerken dazu:

> "Änderungen an Materialien werden häufig in privaten Lokalitäten, die nicht von anderen Akteuren betreten oder eingesehen werden können, vorgenommen. Somit wird ein Höchstmaß an Handlungssicherheit gewährleistet. Der „Besitzer" der privaten Lokalität kann davon ausgehen, daß sich die Materialien in der privaten Lokalität nur in der von ihm initiierten Weise ändern."[232]

Die Abteilungsleiterin schreibt z.B. das Mailing in eigener Verantwortung und möchte daher nicht, dass ihre Mitarbeiterinnen während des Schreibens darauf zugreifen können. Diesen Wunsch nach Handlungssicherheit setzt sie um, indem sie das Mailing an ihrem privaten Arbeitsplatz bearbeitet.

An Einzelarbeitsplätzen ist ein Höchstmaß an Handlungssicherheit oft die Regel. Die Möglichkeiten an gemeinsamen Plätzen – insbesondere die Zugriffsmöglichkeiten – verringern jedoch die Handlungssicherheit oder heben sie sogar vollkommen auf. So können an gemeinsamen Plätzen mehrere Benutzer Dinge unvermutet verändern. Auf der anderen Seite schränkt ein Höchstmaß an Handlungssicherheit die Möglichkeiten zur Kooperation massiv ein oder verhindert sie sogar.
Beides ist aber für "qualifizierte und eigenverantwortliche Tätigkeiten"[233] unverzichtbar. Somit muss es neben einer öffentlichen gemeinsamen Umgebung auch eine vollkommen private Umgebung geben, in der vollkommene Handlungssicherheit herrscht – dies ist zumeist der private Arbeitsplatz.

Darüberhinaus weisen Roock und Wolf darauf hin, dass Personen Materialien von einem Platz zum anderen tragen können, z.B. in der Hand, Hosentasche, Handtasche, Koffer usw. Eine Person, die ein Material bei sich führt ist jederzeit in der Lage, auf das Material zuzugreifen und es einzusehen oder zu bearbeiten. So kann man sich leicht vorstellen, dass die Abteilungsleiterin z.B. die Notiz zur Kontrolle noch einmal durchliest, während sie sie zusammen mit dem Mailing von ihrem Arbeitsraum ins Sekretariat trägt. Andere Personen können währenddessen nicht auf die mitgeführten Materialien zugreifen. Die mitgeführten Materialien befinden sich – genau wie die Dinge am Arbeitsplatz – in einer vollkommen privaten Sphäre. Roock und Wolf nennen diesen Platz, der einer Person "anhaftet", den *Privatgebrauch* der Person:

> "Jede Person kann Gegenstände in ihrem Privatgebrauch mit sich führen und sie in diesem sondieren und bearbeiten. Der Privatgebrauch haftet der Person an und kann daher als „bewegliche private Lokalität" aufgefaßt werden."[234]

---

[232] [RoockWolf98], S.32. "Lokalitäten" sind hierbei gegeneinander abgrenzbare Orte, an denen Handlungen durchgeführt werden (S.30).
[233] siehe WAM-Leitbild, Kap.1.1.3, S.18
[234] [RoockWolf98], S.33

Der private Arbeitsplatz und der Privatgebrauch bilden gemeinsam die private Umgebung einer Person, die der gemeinsamen Umgebung aller beteiligten Personen gegenübersteht. Zwischen diesen Umgebungen kann die Person, um deren private Umgebung es sich handelt, Dinge unmittelbar austauschen. So kann z.B. die Abteilungsleiterin die Notiz einfach aus der Hand legen und wieder aufnehmen. Dabei ist der handelnden Person bewusst, welcher Teil ihrer Umgebung privat ist (z.B. ihr Arbeitsplatz, ihre Hände usw.) und welcher nicht. Auf diese Weise kann jede Person entscheiden, ob sie Dinge exklusiv benutzen oder anderen Personen zur Verfügung stellen möchte, indem sie sie an den entsprechenden Platz legt.

In vielen Fällen ist die vollkommene Handlungssicherheit der privaten Umgebung jedoch zu strikt. So ist z.B. das Arbeitszimmer der Abteilungsleiterin ein privater Bereich, jedoch mit entscheidenden Einschränkungen. Es entspricht z.B. durchaus den Konventionen, dass eine Mitarbeiterin wie im ersten Szenario eine Notiz in diesem Arbeitszimmer ablegt. Auch kann es sein, dass eine Mitarbeiterin ein Buch aus dem Arbeitszimmer entleiht.
Ein solcher Platz ist aufgrund dieser Zugriffsmöglichkeiten natürlich unbestreitbar Teil der gemeinsamen Umgebung. Hier darf die vollkommene Handlungssicherheit nicht prinzipiell garantiert werden, sondern kann durch andere Mittel in abgeschwächterer Form realisiert werden. Dafür gibt es zahlreiche Möglichkeiten, die von bloßen Zugriffskonventionen oder der Überwachung von Plätzen bis hin zu Schlössern an Räumen, Schreibtischen und anderen Behältern[235] reichen. Im Falle des Arbeitszimmers der Abteilungsleiterin gelten lediglich bestimmte Konventionen, die die Handlungen der anderen Mitarbeiterinnen in diesem Raum einschränken. So ist es unüblich, ein Mailing unaufgefordert vom Schreibtisch der Abteilungsleiterin zu nehmen und zu verschicken – selbst wenn die entsprechende Notiz dabei liegen sollte.

Konzept	Zweck
private Umgebung	für ein Höchstmaß an Handlungssicherheit
private Arbeitsplätze	um Arbeiten eigenverantwortlich mit der dafür notwendigen Handlungssicherheit verrichten zu können
Privatgebrauch	um Dinge bei sich führen und jederzeit darauf zugreifen zu können
eingeschränkter Zugriff auf die Plätze der gemeinsamen Umgebung	für Handlungssicherheit in unterschiedlich starkem Maße
eine Person kann Dinge zwischen ihrer privaten Umgebung und der gemeinsamen Umgebung austauschen	um zwischen vollkommener Handlungssicherheit und der Möglichkeit zur Kooperation frei wechseln zu können

Tabelle 5: Konzepte zum Problem "Handlungssicherheit"

[235] [Beis00] zum Thema Zugriffskontrolle durch Schlösser

3. Entwurf der COJAC-Raumkomponente

Schwerpunkt dieses Kapitels ist der Entwurf einer Basis-Raumkomponente, die den im vorangegangenen Kapitel ausgearbeiteten Anforderungen genügt.

Auf dem Entwurf der Basis-Raumkomponente aufbauend, möchte ich anschließend eine erweiterte Raumkomponente entwerfen. Dies soll einen Ausblick auf die mögliche Weiterentwicklung geben. Realisiert habe ich im Rahmen dieser Arbeit die Basis-Raumkomponente[236].

Das Kapitel gliedert sich in vier Teile. Bevor ich zum eigentlichen Entwurf komme, betrachte ich im ersten Teil zunächst den Begriff der virtuellen Arbeitsumgebung. Unter dem Aspekt der Kooperationsunterstützung ergibt sich ein differenzierteres Verständnis der WAM-Entwurfsmetaphern Arbeitsumgebung und Arbeitsplatz. Das Konzept des Einzelarbeitsplatzes arbeite ich weiter aus und stelle ihn als private Umgebung neben eine gemeinsame Umgebung.

Nach diesen grundsätzlichen Vorüberlegungen entwerfe ich im zweiten Teil dieses Kapitels das fachliche Modell[237] der Basis-Raumkomponente. Darin werden die relevanten Konzepte und Gegenstände eines Anwendungsbereichs und der spezifische Umgang mit diesen beschrieben. Das fachliche Modell steht, wie beschrieben, in sehr engem Zusammenhang zum technischen Modell[238], das im nächsten Hauptkapitel im Zusammenhang mit der Konstruktion der Basis-Raumkomponente entworfen wird.

Im dritten Teil des Kapitels entwerfe ich das Benutzungsmodell[239] in Form einer detaillierten Systemvision. Die Systemvision ist gleichzeitig eine Beschreibung der tatsächlich realisierten Basis-Raumkomponente[240].

Am Ende dieses Kapitels, im vierten Teil, steht schließlich der Ausblick auf eine mögliche erweiterte Raumkomponente.

[236] zur Beschreibung der Konstruktion, siehe Kap.4, S.119
[237] Kap.1.1.2, S.15
[238] Kap.1.1.2, S.15
[239] Kap.1.1.2, S.16
[240] zu den Gründen für diese Verknüpfung siehe Einleitung zum Kap.3.3, auf S.89

3.1 Eine virtuelle Arbeitsumgebung für die kooperative Arbeit

Alle Kooperationspartner haben unterschiedliche, individuelle Sichten auf ihre virtuelle Arbeitsumgebung. Bevor diese entworfen werden kann, muss zunächst geklärt werden, wie die individuellen Sichten mit einem gemeinsamen Verständnis zusammenpassen.

Der WAM-Ansatz mit seinem Leitbild des *Arbeitsplatzes für eigenverantwortliche Expertentätigkeit* ist ursprünglich auf die Entwicklung von Einzelarbeitsplätzen fokussiert. Die Rolle des Einzelarbeitsplatzes verändert sich jedoch bei der Unterstützung kooperativer Arbeit. In [WAM98] werden daher *Arbeitsumgebung* und *Arbeitsplatz* unterschieden:

Begriffe 17 und 18: Arbeitsumgebung, Arbeitsplatz

"Die *Arbeitsumgebung* ist der Ort, wo Werkzeuge, Materialien und andere Gegenstände, die bei der Erledigung von Aufgaben griffbereit sein müssen, fachlich motiviert angeordnet sind. Dabei findet die eigentliche Arbeit am *Arbeitsplatz* statt, während zur *Umgebung* noch die Orte gehören, die unmittelbar zugänglich sind.

Der (individuelle) *Arbeitsplatz* ist gegen den Zugriff von außen geschützt. Wenn nur die Arbeit eines einzelnen Benutzers unterstützt werden soll, fallen Arbeitsplatz und –umgebung meist zusammen."[241]

Die eigentliche Arbeit wird am Arbeitsplatz verrichtet. Dort werden Werkzeuge benutzt, in der restlichen Umgebung nicht. Natürlich können sich Werkzeuge auch in der restlichen Umgebung befinden – sie haben dann jedoch Materialcharakter, d.h sie können dort nicht zur Bearbeitung anderer Materialien benutzt werden. Dies ist eine grundlegende Unterscheidung zwischen individuellem Arbeitsplatz und Umgebung.

Im Gegensatz zur restlichen Umgebung ist der individuelle Arbeitsplatz ein strikt privater Platz, an dem für den Benutzer ein Höchstmaß an Handlungssicherheit gilt. Wie bereits im vorhergehenden Kapitel besprochen, ist diese strikte Privatheit Voraussetzung für eigenverantwortliches Handeln[242]. Ohne diese Privatsphäre wäre jede Handlung öffentlich und könnte somit zum Gegenstand von Kritik werden – dies würde den Benutzer in seiner Handlungsweise verunsichern[243].

Aus der Sicht eines einzelnen Benutzers teilt sich die Umgebung also in zwei Bereiche: den individuellen Arbeitsplatz (die *private Umgebung* des Benutzers) und einen öffentlichen Bereich (die *gemeinsame Umgebung* aller Benutzer). Die individuellen Arbeitsplätze der übrigen Benutzer gehören nicht zu der Umgebung dieses Benutzers. Das bedeutet, dass sich die Umgebungen der verschiedenen Benutzer unterscheiden – ich nenne sie daher die *individuellen Umgebungen* der Benutzer.

---

[241] [WAM98] S.183
[242] Kap.2.2.3, S.65
[243] [RoockWolf98] S.33

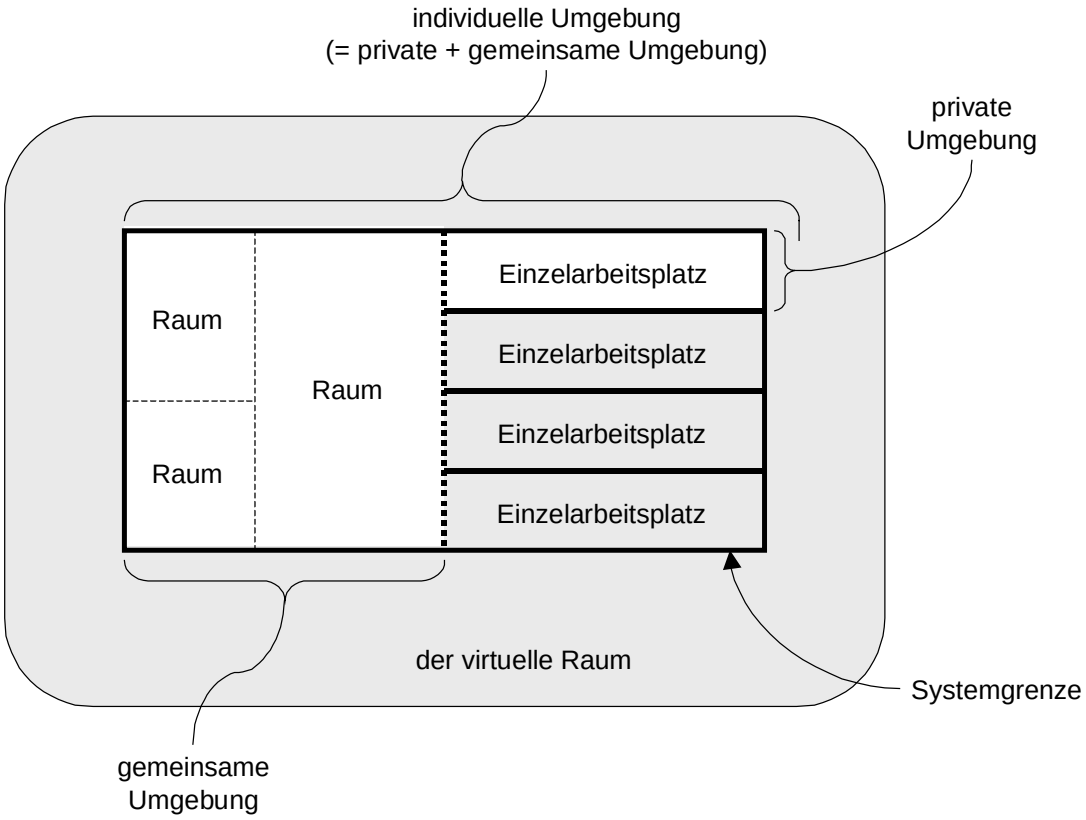


Abbildung 10: Der virtuelle Raum, Umgebungen, Plätze

Am individuellen Arbeitsplatz befinden sich persönliche oder private Dinge[244] eines Benutzers und alle anderen Dinge, die er exklusiv nutzen möchte. Einige dieser Dinge benötigt der Benutzer ständig bei sich (z.B. Kugelschreiber, Notizblock, Schlüssel, usw.)[245], andere trägt er zeitweise bei sich, um sie von einem Platz zum anderen zu transportieren. Diese Dinge befinden sich in seiner unmittelbaren Privatsphäre, also gewissermaßen an einem strikt privaten Platz, der ihm "anhaftet", dem Privatgebrauch[246].

Die Begriffe "persönlich" und "privat" verwende ich in dieser Arbeit nicht synonym, sondern mit der folgenden Bedeutung:

[244] siehe weiter unten folgende Definition der Begriffe "persönlich" und "privat"
[245] In einem WAM-System zählen dazu insbesondere auch solche Werkzeuge, die dem Benutzer den grundlegenden Umgang mit den Materialien in seiner Umgebung ermöglichen. Um z.B. eine Liste durchzusehen, benötigt der Benutzer eine Art Auflister-Werkzeug – obwohl er in der physischen Welt für derartige Tätigkeiten kein Werkzeug, sondern lediglich seine Hände und einige seiner fünf Sinne verwendet.
[246] Kap.2.2.3, S.66

Begriffe 19 und 20: persönlich, privat

Der Begriff "persönlich" bedeutet "einer Person zugehörig". Der Begriff "privat" steht für "nicht öffentlich".

Auf die persönlichen Dinge eines Benutzers können andere Benutzer u.U. zugreifen, auf die privaten Dinge nicht. Welche Dinge persönlich sind, wird durch Konventionen bestimmt, welche privat sind, bestimmt der jeweilige Benutzer.

Privatgebrauch und Arbeitsplatz sind zwei unterschiedliche Plätze, obgleich beide strikt privat sind. Der entscheidende Unterschied ist dabei, dass der Privatgebrauch beweglich ist und ständig zugreifbar, der individuelle Arbeitsplatz (bei der Büroarbeit i.d.R. ein Schreibtisch) jedoch nicht. Entfernt sich z.B. der Benutzer von seinem Arbeitsplatz, so kann er nicht mehr auf diesen Teil seiner privaten Umgebung zugreifen. Das ist sicher eine durchweg nachteilige Einschränkung, da dort die eigentliche Arbeit stattfindet[247].
Auf der anderen Seite kann der Benutzer unmöglich alle Dinge, die er zur Erledigung seiner Aufgaben bereithalten muss, bei sich tragen. Es entspricht nicht unserer Vorstellung von einem Büroarbeitsplatz, dass ein Benutzer diesen – quasi wie einen Bauchladen – mit sich herumträgt. Dementsprechend unsinnig erscheint es, diese beiden Plätze im Entwurf eines Softwaresystems zu vereinen.

In Bezug auf ein Softwaresystem stellt sich die Situation allerdings in entscheidender Weise anders dar als in der physischen Welt. Innerhalb der physischen Welt bewegt sich der Benutzer zwar umher, auf die virtuelle Welt schaut er jedoch "von außen". Schließlich befindet er sich vor seinem physischen Computerarbeitsplatz, von dem aus er auf die virtuelle Welt zugreift. Die im WAM-Ansatz angestrebte Strukturähnlichkeit zwischen Software und Anwendungsbereich[248] ist somit in diesem Punkt ausgeschlossen. Hier muss ein neues Konzept das Betreten eines Platzes ersetzen.

3.1.1 Fokussieren eines virtuellen Platzes

Wenn sich ein Benutzer an einen virtuellen Platz "begibt", so bedeutet das, dass er diesen Platz nicht wirklich betritt, sondern ihn lediglich in irgendeiner Form mithilfe eines Werkzeugs fokussiert[249]. Einen Platz zu fokussieren soll bedeuten, diesen Platz genauer zu betrachten, auf die Dinge an diesem Platz zuzugreifen und (an Plätzen der gemeinsamen Umgebung) von anderen Benutzern wahrgenommen zu werden. Insoweit entspricht das Fokussieren eines virtuellen Platzes dem Betreten eines physischen Platzes. Das beim Fokussieren bzw. Betreten ausgenutzte Merkmal einer räumlichen Umgebung bezeichnen Harrison und Dourish als presence and awareness[250].

---

[247] Kap.3.1, Begriffe 17 und 18, S.69
[248] [WAM98], S.263
[249] Die Form des Fokussierens hängt dabei von dem Werkzeug ab, welches den virtuellen Platzes präsentiert. Die Frage der Form soll bei diesen Betrachtungen keine Rolle spielen und erst im Benutzungsmodell (Kap.3.3.2, S.94) wieder aufgegriffen werden.
[250] Kap.1.3.5, S.39

Das Fokussieren ist eine Einstellung am Werkzeug, das den jeweiligen Platz visualisiert. Der Platz selbst wird durch das Fokussieren nicht verändert. Es erscheint mir daher nicht sinnvoll, das Fokussieren eines virtuellen Platzes als eine besondere Umgangsform im fachlichen Modell zu berücksichtigen. Das Konzept des Fokussierens wird erst im Benutzungsmodell konkretisiert.

3.1.2 Die Rollen von individuellem Arbeitsplatz und Privatgebrauch

Offenbar ist damit ist die Eigenschaft des Privatgebrauchs, dem Benutzer anzuhaften und sich mit ihm zwischen den Plätzen umherzubewegen, irrelevant geworden. Entscheidend ist vielmehr, dass der Privatgebrauch ständig zugreifbar ist – unabhängig davon, welchen Platz der Benutzer gerade in der Umgebung fokussiert.

Das ist auch für den individuellen Arbeitsplatz eine sehr wünschenswerte Eigenschaft. Da dort die eigentliche Arbeit stattfindet, ist – wie gesagt – nicht einzusehen, warum er für den Benutzer vorübergehend nicht zugreifbar sein sollte – z.B. wenn er auf die übrige Umgebung zugreift.
Die strikte Privatheit des individuellen Arbeitsplatzes bedeutet, dass andere Benutzer prinzipiell nicht darauf zugreifen können. Somit benötigen diese anderen Benutzer keinen Zugriffspunkt darauf. Es ist also unnötig, für den individuellen Arbeitsplatz einen festen Standort innerhalb der gemeinsamen virtuellen Umgebung vorzusehen.

Dieses Verständnis des individuellen Arbeitsplatzes mag für manche Anwendungsfälle zu strikt und einseitig erscheinen. Mitunter ist es z.B. üblich, einem Kollegen eine Notiz auf seinen Schreibtisch zu legen, also – der geforderten Handlungssicherheit zum Trotz – auf dessen Arbeitsplatz zuzugreifen.
In diesem Fall sind der Schreibtisch oder Teile davon (z.B. die Schreibtischoberfläche) Teil der gemeinsamen Umgebung. Der Schreibtisch ist also als besonderer Platz innerhalb der gemeinsamen Umgebung zu modellieren[251]. Daneben steht jedoch das Modell des strikt privaten, individuellen Arbeitsplatzes, welches weiterhin benötigt wird. Hier entsteht ein gewisser Bruch zwischen Analyse und Entwurf, da der physische Arbeitsplatz auf zwei virtuelle Plätze aufgeteilt wird: den virtuellen Schreibtisch in der gemeinsamen Umgebung und strikt privaten Arbeitsplatz. Die klare Trennung zwischen privater und gemeinsamer Umgebung erscheint mir wichtiger, als die genaue Nachbildung des Arbeitsplatzes in Form des Schreibtisches, daher denke ich, dass der Bruch vetretbar ist.

Es sind Systeme denkbar, in denen die Handlungssicherheit eine untergeordnete Rolle spielt oder in dem hier geforderten Höchstmaße sogar unerwünscht ist, z.B. bei der Unterstützung von Gruppenarbeitsplätzen[252] oder in Systemen zur EDV-Schulung[253]. Um solche Systeme zu

[251] siehe Kap.3.4.2, S.107 zur Möblierung von virtuellen Räumen
[252] An Gruppenarbeitsplätzen können situationsbedingt Arbeitsschritte, die von einer Person begonnen wurden, von einer anderen Person fortgeführt werden. Daher ist der freie Zugriff auf alle Arbeitsmaterialien erforderlich. ([WAM98], S.928). Hier ist nur ein Minimum an Handlungssicherheit erwünscht.

realisieren, ist z.B. denkbar, dass der Arbeitsplatz Dinge nur temporär enthält, solange sie von einem Benutzer bearbeitet werden. Währenddessen wird über Awarenessinformationen öffentlich gemacht, welcher Benutzer mit welchem Werkzeug auf welchem Material arbeitet. Nach der Benutzung werden die verwendeten Dinge automatisch an ihren Platz in der gemeinsamen Umgebung zurückgelegt. Der Benutzer hat keine Möglichkeit, Dinge dauerhaft an seinem privaten Arbeitsplatz aufzubewahren und so vor dem Zugriff durch andere Benutzer zu verbergen.
Das ist jedoch ein vollkommen anderes Konzept des individuellen Arbeitsplatzes. Die typische Handlungssicherheit des traditionellen Einzelarbeitsplatzes – in seiner Umsetzung als Einzelplatzsystem – wird hier nicht aufrechterhalten.

In dieser Arbeit gehe ich konservativ vor und versuche, ein Konzept einer gemeinsamen Umgebung neben dem bestehenden Konzept des Einzelarbeitsplatzes zu etablieren.
Das Konzept des traditionellen Einzelarbeitsplatzes hat sich in der Vergangenheit bewährt und auch in einem kooperativen Kontext seine Berechtigung.
Zudem wird dieses Konzept im JWAM-Framework bereits durch eine leistungsfähige Desktop-Komponente unterstützt, die im Rahmen von [Lippert99] realisiert wurde. Mit Blick auf das Ziel dieser Arbeit – der Konstruktion einer Raumkomponente für das JWAM-Framework – liegt nahe, die hier entwickelten Konzepte, mit den Konzepten des JWAM-Desktops zu verbinden.
Im JWAM-Framework vergegenständlicht der Desktop die Arbeitsumgebung[254] – dabei muss berücksichtigt werden, dass der JWAM-Desktop zur Unterstützung der Arbeit eines einzelnen Benutzers entwickelt wurde[255]. Im kooperationsunterstützenden Kontext vergegenständlicht der Desktop nur einen Teil der individuellen Arbeitsumgebung, nämlich die private Umgebung.
Das fachliche Konzept des JWAM-Desktops hat in diesem Sinne weiterhin bestand und ich werde mich im Entwurf der Raumkomponente darauf berufen[256].

Anderen Konzepten des Einzelarbeitsplatzes, wie z.B. zuvor beschrieben, gehe ich hier nicht weiter nach.

Zwischen Privatgebrauch und individuellem Arbeitsplatz besteht in der virtuellen Umgebung kein entscheidender Unterschied mehr: Beide Plätze sind für ihren "Besitzer" permanent zugreifbar und für die übrigen Benutzer prinzipiell nicht zugreifbar. Aus diesem Grund fasse ich sie zu einem einzelnen Platz zusammen. Diesen Platz nenne ich *Einzelarbeitsplatz* und sehe ihn in der Tradition des Einzelarbeitsplatzes, der einem nicht kooperations-unterstützenden Kontext entstammt.

---

[253] In Systemen zur EDV-Schulung kann es notwendig sein, dass ein Lehrer auf den Arbeitsplatz eines Schülers zugreift, um ihm z.B. Hilfestellung zu geben, ihm etwas zu demonstrieren usw. Die Handlungssicherheit ist für einen Schüler dabei minimal.
[254] [Lippert99] S.28f.
[255] [Lippert99] S.29
[256] die Umgangsformen mit einem Einzelarbeitsplatz finden sich in Kap.3.2.3.1, Abb.12, S.82

In einem kooperationsunterstützenden Kontext vergegenständlicht der Einzelarbeitsplatz die private Umgebung eines Benutzers. Das Konzept des Einzelarbeitsplatzes übernimmt und vereinigt dabei die Rollen der beiden Konzepte Privatgebrauch und individueller Arbeitsplatz.

3.1.3 Die räumliche Beziehung von Einzelarbeitsplatz und gemeinsamer Umgebung

Die individuelle Umgebung eines Benutzers erstreckt sich also über dessen Einzelarbeitsplatz (private Umgebung) und die gemeinsame Umgebung.

Den Einzelarbeitsplatz stelle ich im virtuellen Raum neben die gemeinsame Umgebung (siehe Abb.10), da beide Konzepte – trotz ihres Zusammenspiels und ihrer Gemeinsamkeiten – voneinander unabhängig sind. Letztlich ist es aber eine offene fachliche Frage, ob sich der virtuelle Einzelarbeitsplatz innerhalb oder außerhalb der gemeinsamen Umgebung befindet.

Bei der Konzeption der gemeinsamen virtuellen Umgebung darf jedoch nicht vergessen werden, dass beim Einsatz des Systems – so sehr die Anwendungsorientierung im Vordergrund steht – die technische Realität sich auf die Handhabung auswirkt. Deutlich wird dies z.B. in einer verteilten Umgebung, in der Verbindungsprobleme auftreten können. Dem Benutzer muss ermöglicht werden, die für ihn spürbaren technischen Bedingungen zu verstehen, um z.B. in Problemfällen angemessen reagieren zu können.

Die technische Realisierung der Umgebungen in einer verteilten Architektur kann also gute fachliche Gründe liefern, sich für den Fall zu entscheiden, dass sich der Einzelarbeitsplatz außerhalb der gemeinsamen Umgebung befindet. Die Grundlage hierfür bildet die im WAM-Ansatz geforderte *fachliche Transparenz*:

> "[Fachliche Transparenz] bedeutet, daß Verteilung bei der Kooperation sichtbar also durchschaubar sein soll. Wir wollen dem Anwender (und auch dem Anwendungsentwickler) ein nachvollziehbares Bild von begrenzten Ressourcen und ihrer räumlichen Verteilung geben, um den Aufwand und die Folgen eines Zugriffs auf nicht-lokale Komponenten zu verdeutlichen."[257]

Befindet sich der Einzelarbeitsplatz z.B. auf einem lokalen Rechner, die gemeinsame Umgebung jedoch auf einem entfernten Server, so kann dies auf der fachlichen Ebene ausgedrückt werden, indem private und gemeinsame Umgebung nebeneinander gestellt werden. Die technische räumliche Trennung der Umgebungen findet sich dann auch im fachlichen Modell wieder.

Die technische Verteilung der Umgebungen kann jedoch nicht mit einem fachlichen Modell zur Deckung gebracht werden, in dem sich der Einzelarbeitsplatz innerhalb der gemeinsamen Umgebung befindet.
Ist eine Umgebung nicht erreichbar, sind damit auch die darin enthaltenen Dinge und Umgebungen nicht erreichbar. Ist die gemeinsame Umgebung nicht erreichbar, z.B. aufgrund eines Verbindungsproblems, müsste dies gemäß dem fachlichen Modell auch für die private

[257] [WAM98], S.882

Umgebung gelten, da diese in der gemeinsamen Umgebung enthalten ist. Das steht aber im Widerspruch zur technischen Realität. Da der Benutzer "von außen" auf die Umgebungen schaut, wäre es für ihn nicht nachvollziehbar, warum er die enthaltene private Umgebung weiterhin erreichen kann, die gemeinsame Umgebung, in der die private enthalten ist, jedoch nicht.

Soweit ich es bis jetzt beurteilen kann, gibt die geforderte fachliche Transparenz den Ausschlag für ein fachliches Modell, in dem private und gemeinsame Umgebung nebeneinander stehen.

3.2 Das fachliche Modell

Die Frage ist nun, wie eine gemeinsame Umgebung konkret zu gestalten ist: Aus welchen Teilen besteht sie? Wie werden diese Teile gehandhabt? Die erste Frage beantwortet das fachliche Modell, die zweite (im Kapitel 3.3) das Benutzungsmodell.

Den Entwurf des fachlichen Modells der Raumkomponente beschreibe ich anhand des Beispiels aus dem Buchverlag. Dazu untersuche ich die in den Szenarios beschriebene Arbeitsumgebung im Folgenden näher: Was enthält sie? Wer arbeitet in ihr? Wie ist sie aufgebaut?
Neben den Ressourcen sind es die zu repräsentierenden Personen und die Plätze, die es zu modellieren gilt. Für die Ressourcen und Personen überprüfe ich zunächst im Einzelnen, inwieweit sie für das fachliche Modell relevant sind, um sie ggf. anschließend zu modellieren. Bei der Modellierung der Plätze beziehe ich mich auf die bereits im zweiten Kapitel ausgearbeiteten Anforderungen und Konzepte, die es mir ermöglichen, bereits ein generischeres fachliches Modell zu entwerfen.

Die folgende Grafik zeigt die Umgebung der PR-Abteilung aus dem Beispiel Buchverlag.

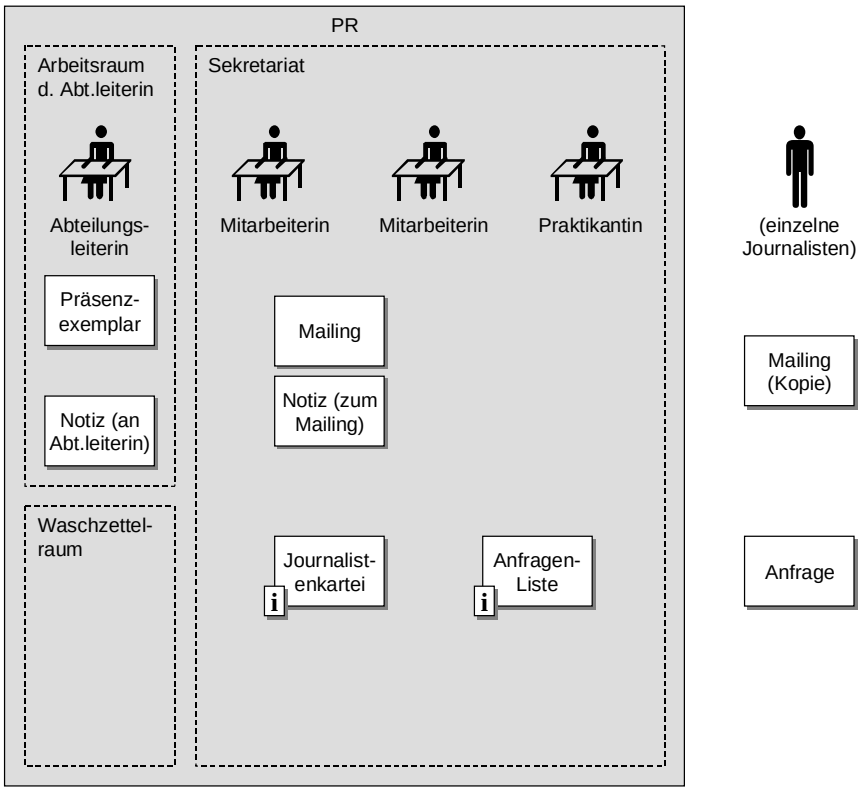


Abbildung 11: Personen, Plätze und Ressourcen in den Beispiel-Szenarios

Hier sind – wie in den Kooperationsbildern aus Kapitel 2 – die vier Mitarbeiterinnen an ihren Arbeitsplätzen innerhalb der Räumlichkeiten der PR zu erkennen. Außerhalb der Umgebung der PR befinden sich die Journalisten. Zudem sehen wir die verwendeten Ressourcen – mit einem "i" für "Inventar" markiert, sofern sie sich an ihrem Stammplatz befinden.

3.2.1 Personen

Bei den Personen aus den Szenarios handelt es sich um die vier Mitarbeiterinnen der PR-Abteilung und – außerhalb der Umgebung der PR – die verschiedenen Journalisten.

Die vier Mitarbeiterinnen der PR sind im Einzelnen:
- die Abteilungsleiterin,
- die zwei Mitarbeiterinnen des Sekretariats,
- die Praktikantin.

Sie sollen auf ihre virtuelle Arbeitsumgebung zugreifen können[258] und sich in der gemeinsamen Umgebung dabei gegenseitig wahrnehmen (Awareness)[259].

Um Awareness zu ermöglichen, muss der einzelne Benutzer identifiziert werden können – dies ist leicht durch einen speziellen Fachwert[260] zu modellieren. Mithilfe dieses Fachwerts können die Handlungen eines Benutzers quasi "signiert" werden.
Das Benutzerkonzept wird in [Beis00] unter dem Aspekt der Zugriffskontrolle weiter ausgearbeitet. Darauf werde ich in der Vision der erweiterten Raumkomponente wieder zurückkommen[261]. Für den Entwurf der Basis-Raumkomponente möchte ich mich jedoch vorerst mit der minimalen Lösung, einen Benutzer über einen Fachwert zu identifizieren, begnügen. Der Fachwert bietet dafür eine String-Repräsentation, die z.B. den Namen des Benutzers enthalten kann.

Im Gegensatz zu den Mitarbeiterinnen der PR arbeiten die Journalisten weder in der Umgebung der PR, noch können sie auf Ressourcen in der Umgebung zugreifen oder die Umgebung einsehen. Ihre Handlungen müssen demnach nicht mit einem entsprechenden Fachwert "signiert" werden. Awareness ist in Bezug auf die Journalisten nicht notwendig.
Sie treten außerhalb der zu modellierenden Umgebung als explizite Kooperationspartner in Erscheinung. Kooperationsmedien sind dabei Telefon, Fax und Post. Die Frage, wie die Journalisten modelliert werden sollten, hängt u.a. davon ab, inwieweit die explizite Kooperation mit den Journalisten unterstützt werden soll. Die Journalistenkartei enthält z.B. die für die Kommunikation via Post, Fax und Telefon notwendigen Telefonnummern und Adressen. Die Journalisten können also gut durch die Einträge in der Journalistenkartei repräsentiert werden. Auf die Journalistenkartei komme ich im folgenden Unterkapitel zurück.

[258] Kap.2.2.1, S.61 und Kap.2.2.3, S.65
[259] Kap.2.2.2, S.64
[260] Kap.1.1.3, S.22
[261] Kap.3.4.3, S.110

3.2.2 Ressourcen

Einen wesentlichen Teil des fachlichen Modells des Buchverlags machen die in den Szenarios genannten Ressourcen aus. Diese sind:

- das Präsenzexemplar des neuen Buches,
- das Mailing,
- die Notiz zum Mailing,
- die zu verschickenden Kopien des Mailings,
- die Notiz an die Abteilungsleiterin, dass die Mailing-Aktion durchgeführt wurde,
- die Journalistenkartei,
- die Anfragen der Journalisten nach Musterexemplaren,
- die Anfragenliste[262].

Es stellt sich die Frage, für welche Gegenstände der Liste es sinnvoll ist, sie tatsächlich zu modellieren.

Die Autorin des Mailings benötigt auf jeden Fall das gedruckte Präsenzexemplar des neuen Buches. Eine Datei mit dem bloßen Buchtext wird ihr nicht genügen, da sie sich auch vom Gesamteindruck des Buches inspirieren lässt – das Buch als "sinnliches Erlebnis". Dies wird sich in Zukunft vermutlich ändern, falls die neu aufgekommenen elektronischen Bücher (oder kurz E-Books)[263] eine größere Rolle spielen – allerdings wird sich dann sicher auch der Buchverlag in seiner Organisationsstruktur verändern. Für den betrachteten "traditionellen" Buchverlag wird das Präsenzexemplar im gedruckten und gebundenen Original benötigt und ist somit kein Kandidat für das fachliche Modell.

Das Mailing ist dagegen ein eindeutiger Kandidat für das fachliche Modell.
Die Kopien der Mailings werden im Szenario per Post verschickt, dazu wäre es nicht nötig, die Kopien zu modellieren. Ich denke aber, dass ein elektronisches Postsystem verwendet werden kann, da dieses Medium in Form von E-Mail allgemein verbreitet ist – also auch bei den adressierten Journalisten. Das COJAC-Postversandsystem[264] kann dahingehend erweitert werden, dass es zum Versand auf E-Mail aufsetzt. Daher sehe ich es hier für den elektronischen Postversand vor.
Die Kopien des Mailings berücksichtige ich im fachlichen Modell.

Durch die Einbindung des COJAC-Postversandsystems muss das fachliche Modell um die dazugehörigen Postkörbe erweitert werden. Es ist zu überlegen, ob die Postkörbe besser allen gemeinsam zugänglich sind oder ob jede Mitarbeiterin über eigene Postkörbe verfügt. Ich habe mich für den letzteren Fall entschieden und betrachte die Postkörbe als Ausstattung der

---

[262] Da nur ein Ausschnitt der PR eines Buchverlags modelliert wird, fehlen in dieser Liste natürlich sehr viele Ressourcen, wie z.B. Waschzettel, Lieferscheine, Fotos, Kataloge, Pressespiegel usw. Für diese Arbeit soll der gewählte Ausschnitt genügen.
[263] Einige Buchwerke können seit kurzem als E-Book-Editionen aus dem Internet heruntergeladen und mit einem kleinen, buchähnlichen Lesecomputer mit LC-Bildschirm – fast wie ein gebundenes Buch – gelesen werden. Oft wird dem E-Book vorgeworfen, dass es kein "sinnliches Erlebnis" vermittelt.
[264] [Freund00]

Einzelarbeitsplätze. Auf diese Weise können die Mitarbeiterinnen neben der impliziten Kooperation über die gemeinsame Umgebung auch explizit über das Postversandsystem kooperieren[265].

Auf den Einträgen der Journalistenkartei finden sich u.a. Name, Adresse und Telefonnummer der Journalisten. Es erscheint mir konsequent, auch die Adressen für das COJAC-Postversandsystem dort zu verwalten.
Die Journalistenkartei gehört sinnvollerweise in das fachliche Modell. Für die Journalistenkartei gibt es weit verbreitete kommerzielle Softwarelösungen – die Frage der Einbettung soll bei meinen weiteren Betrachtungen jedoch keine Rolle spielen. Der Einfachheit halber nehme ich an, dass die Journalistenkartei mithilfe der JWAM-Registratur[266] realisiert wird.
Die Journalistenkartei hat ihren festen Stammplatz im Sekretariat, sie gehört zum Inventar des Raumes.

Die Anfragen der Journalisten werden telefonisch oder per Fax übermittelt. Dazu müssen sie nicht modelliert werden. Selbst wenn die Journalisten E-Mail verwenden sollten, so handelt es sich bei der Anfrage nicht um ein spezielles Material im WAM-Sinn, da die Anfrage z.B. ein Abschnitt in einer Mail zu einem anderen Thema sein kann. Es wäre nicht angemessen, die Anfrage zu formalisieren. Sie wird also nicht modelliert.
Die Anfragenliste wird jedoch modelliert[267].

Die beiden noch nicht diskutierten Gegenstände, die Notiz zum Mailing und die Notiz an die Abteilungsleiterin, sind meiner Ansicht nach Teil des fachlichen Modells.

Soweit zu den genannten Ressourcen. Im Waschzettelraum, der in den Szenarios nicht genannt wurde, werden alle Waschzettel[268] in Hängeordnern archiviert. Dies kann sehr gut durch die JWAM-Registratur unterstützt werden, deren Stammplatz der Waschzettelraum ist. Dieses "Waschzettelarchiv" gehört zum Inventar des Waschzettelraumes. Ich erweitere das fachliche Modell um dieses Archiv, um auch den Waschzettelraum angemessen auszustatten.

Die Ressourcen im fachlichen Modell sind also nun:
- das Mailing mit den zu verschickenden Kopien,
- die Notiz zum Mailing,
- die Notiz an die Abteilungsleiterin, dass die Mailing-Aktion durchgeführt wurde,
- die Journalistenkartei,
- die Anfragenliste und
- das "Waschzettelarchiv".

[265] Evtl. ist es ein Nachteil, wenn die Postkörbe nicht allgemein zugreifbar sind, zumal in den privaten Postkörben Korrespondenz empfangen wird, die zu einem Teil die gesamte PR-Abteilung gleichermaßen betrifft.
[266] [JWAM00 5]
[267] Auf der Anfragenliste werden die Anfragen u.U. formal notiert – evtl. gibt es ein Material oder einen Fachwert "Anfragenlisteneintrag". Davon sind jedoch nicht die Journalisten betroffen. Würden dagegen die Anfragen explizit modelliert, wären die Journalisten sehr wohl betroffen. Sie müssten dieses Material verwenden und könnten Anfragen nicht mehr formlos z.B. in einer Mail oder am Telefon stellen.
[268] eine Beschreibung der Waschzettel findet sich in Kap.2.1, S.58

3.2.3 Einzelarbeitsplätze und Räume

Die Einzelarbeitsplätze und die Räume stehen im Mittelpunkt dieses fachlichen Modells. Sie machen den Kern der Raumkomponente aus.

Räume sind besondere Orte, sie haben Behältercharakter und ihre Verwendung ist mit Konventionen belegt. Gemäß der Definition des Begriffes "Platz"[269] sind die hier zu modellierenden Räume also Plätze – ebenso wie die bereits in [Lippert99] behandelten Einzelarbeitsplätze.

Die Plätze des Buchverlagsbeispiels sind:
- die vier Einzelarbeitsplätze der Mitarbeiterinnen,
- der Raum der Abteilungsleiterin,
- das Sekretariat und
- der Waschzettelraum.

Weitere Plätze sind in den Szenarios nicht erwähnt worden.
Selbstverständlich sind die Räume der PR möbliert. Somit gibt es eine Vielzahl an weiteren Plätzen innerhalb der Umgebung der PR. An dieser Stelle beschränke ich mich jedoch auf die Räume der PR und betrachte eine mögliche Möblierung erst im Ausblick auf eine erweiterte Raumkomponente[270].

Für die fachliche Modellierung der Plätze habe ich bereits im zweiten Kapitel "Fachliche Motivation der Raumkomponente" verallgemeinerte Anforderungen und Konzepte ausgearbeitet[271]. Diese Vorarbeit gestattet mir nun, für die Plätze ein möglichst generisches fachliches Modell entwerfen. Ein fachliches Modell zu den Beispielen aus dem Buchverlag wäre zu spezifisch, um darauf eine Frameworkkomponente aufzubauen.

Dennoch möchte ich das Beispiel bei der Modellierung nicht übergehen. Daher entwerfe ich mithilfe des generischen Modells das spezielle fachliche Modell zum Buchverlagsbeispiel. Eine Verallgemeinerung verschiedener domänenspezifischer Modelle auf fundierter, empirischer Basis ist kaum zu leisten – den Rahmen dieser Arbeit würde es in jedem Falle sprengen. Aus diesem Grund gehe ich hier den umgekehrten Weg und versuche mit der Modellierung des Beispiels aus dem Buchverlag einen Eindruck von der Verwendbarkeit des generischen Modells zu geben.

Ich beginne mit einer Beschreibung des Modells des Einzelarbeitsplatzes, da dieser, wie im Kapitel 3.1 dargelegt, auch im Zusammenspiel mit der Raumkomponente seine zentrale Bedeutung behält. Es folgen der Raum mit seinen Fachwerten und das Gebäude mit seinem Gebäudeplan.

---

[269] Kap.1.3.6, Begriff 14, S.42
[270] Kap.3.4.2, S.107
[271] Kap.2f., Tabellen 3-5, S.64, S.65 und S.67

Fast alle Teile des fachlichen Modells der Plätze setzen einzelne Konzepte aus dem zweiten Kapitel um. Das umgesetzte Konzept und dessen Zweck stelle ich den einzelnen Teilen jeweils in Form eines Ausschnitts aus den Tabellen aus Kapitel 2 voran.
Ziel ist, daraus die Umgangsformen mit den Plätzen abzuleiten. Dabei greife ich auf das in den Szenarien illustrierte Hintergrundwissen zurück.
Die Umgangsformen und Beziehungen mit dem Einzelarbeitsplatz, dem Raum und dem Gebäude – den zentralen Komponenten dieses fachlichen Modells – fasse ich am Ende des jeweiligen Abschnitts in einer CRC-Karte[272] zusammen.

Zwischen den Abschnitten stehen Anmerkungen zum Entwurf des fachlichen Modells des Buchverlagsbeispiels. Dabei werden jeweils die zuvor entworfenen generischen Komponenten verwendet.

3.2.3.1 Einzelarbeitsplatz

Konzept	Zweck
private Umgebung	für ein Höchstmaß an Handlungssicherheit
private Arbeitsplätze	um Arbeiten eigenverantwortlich mit der dafür notwendigen Handlungssicherheit verrichten zu können
Privatgebrauch[273]	um Dinge bei sich führen und jederzeit darauf zugreifen zu können

Ein Einzelarbeitsplatz vergegenständlicht die private Umgebung eines Benutzers. Dinge können an ihn geholt und wieder von ihm fortgenommen werden. Er hat die Form einer Arbeitsfläche, auf der Dinge an bestimmten Positionen bereitgelegt werden können[274]. Dinge können auf der Arbeitsfläche bewegt und an freien Positionen abgelegt werden.

Die CRC-Karte eines Einzelarbeitsplatzes sieht also folgendermaßen aus:

[272] Kap.1.1.1, S.14
[273] Kap.2.2.3, Tabelle 5, S.67 (alle drei Konzepte)
[274] [Lippert99], S.28

Einzelarbeitsplatz	
ein Ding hineinlegen	**Ding**[275]
prüfen, ob eine Position belegt ist	**Inhaltsverzeichnis**
eine freie Position ermitteln	**Position**
ein Ding an einer best. Position hineinlegen	
ein Ding an eine andere Position bewegen	
prüfen, ob ein Ding herausgenommen werden kann	
ein Ding herausnehmen	
prüfen, ob ein best. Ding enthalten ist	
ein best. Ding ermitteln	
ein Ding an einer best. Position ermitteln	
das Inhaltsverzeichnis ermitteln	
die Position eines best. Dinges ermitteln	

Abbildung 12: CRC-Karte "Einzelarbeitsplatz"

Zum Beispiel aus dem Buchverlag:

Jede der Mitarbeiterinnen hat einen privaten Arbeitsplatz – ihren Einzelarbeitsplatz. Dessen individuelle Ausstattung ermöglicht der jeweiligen Mitarbeiterin, ihre Aufgaben und Verantwortlichkeiten wahrzunehmen. So benötigen die Mitarbeiterinnen offenbar Werkzeuge zum Bearbeiten von Mailings und Notizen.

Um unnötige Konkurrenzsituationen um die Benutzung von Werkzeugen von vornherein auszuschließen, ist es sinnvoll, dass jede Mitarbeiterin ein Exemplar des jeweiligen Werkzeugs auf ihrem Einzelarbeitsplatz hat. Der Ausstattung der individuellen Arbeitsplätze soll hier jedoch nicht weiter nachgegangen werden – ein Beispiel, wie ein solcher Arbeitsplatz ausgestattet sein könnte, findet sich im nachfolgenden Unterkapitel zum Benutzungsmodell[276].

3.2.3.2 Raum

Konzept	Zweck
gemeinsame Plätze[277]	als Zugriffspunkte auf gemeinsame Ressource
unterschiedliche Plätze in der gemeinsamen Umgebung[278]	um die gemeinsame Umgebung und die Arbeit darin strukturieren zu können

[275] Ich verwende hier den Begriff *Ding* statt *Gegenstand* in Anlehnung an den entsprechenden Klassenbezeichner *Thing* aus dem JWAM-Frameworks.
[276] Kap.3.3.1, Abb.15, S.90 – beschrieben auf S.93
[277] Kap.2.2.2, Tabelle 4, S.65
[278] Kap.2.2.1, Tabelle 3, S.64

Räume sind Plätze. Die Anforderung, dass die Plätze gemeinsam genutzt werden, wird erst im Benutzungsmodell umgesetzt, da diese Eigenschaft von der tatsächlichen Verwendung abhängt.

Dinge können in einen Raum hineingelegt und aus ihm herausgenommen werden. Dabei kann ein Raum alle Arten von Dingen enthalten – Ausnahmen sind dabei natürlich Räume und (nach meinem Verständnis[279]) Einzelarbeitsplätze. Die im Raum enthaltenen Dinge werden auf dessen "Fußbodenfläche" an bestimmten Positionen abgelegt. Sie können jederzeit an beliebige freie Positionen verschoben werden.

Die Kapazität eines Raumes ist i.a. unbegrenzt. Es findet sich immer eine freie Position auf seiner Fußbodenfläche. Von der Kapazität eines Raumes wird abstrahiert, da diese in der physischen Welt – wenn sie überhaupt von Bedeutung ist – wohl eher ein Handicap darstellt. Die Ausdehnung dieser Fläche passt sich der Verteilung der darauf befindlichen Dinge an, d.h. sie hängt von der Menge und Anordnung der darauf befindlichen Dinge ab. Gewöhnlich wird von der räumlichen Ausdehnung der Dinge abstrahiert – man denke an die auf 32x32-Pixel normierten Icons des Win95-Desktops. Es wäre unmotiviert, die räumliche Ausdehnung hier wieder einzuführen[280]. Fachlich bedeutsam ist lediglich die räumliche Anordnung der Dinge auf der "Fußbodenfläche" und der Räume im Gebäude – also die Topologie, in der von metrischen Eigenschaften abstrahiert wird.

Durch das physikalische Phänomen der Schwerkraft wird die Ablagemöglichkeit im dreidimensionalen physischen Raum auf zwei Dimensionen – nämlich die der "Fußbodenfläche" – reduziert. Zwar können Dinge gestapelt werden und damit in den dreidimensionalen Raum "hineinragen". Jedoch betrachte ich Stapel als eine besondere Form von Behältern[281], die dann wiederum lediglich auf der zweidimensionalen Grundfläche positioniert werden können. Gleiches gilt für die Anordnung von Dingen in oder auf Möbeln. Somit beschränke ich mich bei der Modellierung der räumlichen Dimensionen eines Raumes auf die zwei Dimensionen seiner "Fußbodenfläche"[282].

Einige der enthaltenen Dinge können zum Inventar des Raumes gehören. Sie erscheinen dann in dessen Inventarliste. Prinzipiell kann jedes beliebige Ding, das in einem Raum enthalten sein kann, auch zu dessen Inventar gemacht werden. Inventarisierte Dinge können nicht aus dem Raum entfernt werden, es sei denn, sie werden zuvor aus dem Inventar entfernt[283].

---

[279] Kap.3.1.3, S.74

[280] Evtl. wäre es sinnvoll, die binäre Größe der Objekte (in Maßen) über ihre Ausdehnung zu visualisieren. Große Objekte erscheinen dann auch groß. Dies ermöglicht dem Benutzer im Sinne der fachlichen Transparenz (Kap.3.1.3, S.74) abzuschätzen, ob das Bewegen eines Objekte zwischen lokaler privater und entfernter gemeinsamer Umgebung zeitaufwendig ist. Diesen Gedanken verfolge ich in dieser Arbeit jedoch nicht weiter.

[281] Kap.3.4.2, S.107

[282] Beim Bewegen von Dingen hat die dritte Dimension eine besondere Bedeutung, da Dinge über die anderen gehoben und bewegt werden können, ohne mit diesen zu kollidieren. Bei der Ablage im Raum kommt die dritte Dimension wie beschrieben jedoch nicht zu Tragen. Man spricht daher oft auch von einem 2½-dimensionalen Raum.

[283] Als Anregung sei hier angemerkt, dass ein Inventarkonzept auch für den Einzelarbeitsplatz sinnvoll sein kann: Damit kann z.B. sichergestellt werden, dass Werkzeuge, mit denen denen spezielle Arbeitsplätze ausgestattet sind, nicht über die Räume der gemeinsamen Umgebung ausgetauscht werden können.

Ein Raum besitzt eine Raumbeschreibung, ein Inhaltsverzeichnis, ein Positionenverzeichnis und eine Inventarliste. Diese teilweise recht komplexen Fachwerte können am Raum sondiert werden.

Raum	
prüfen, ob ein Ding hineingelegt werden kann	**Ding**
ein Ding hineinlegen	**Raumbeschreibung**
prüfen, ob eine Position belegt ist	**Inhaltsverzeichnis**
eine freie Position ermitteln	**Positionenverzeichnis**
ein Ding an einer best. Position hineinlegen	**Inventarliste**
ein Ding an eine andere Position bewegen	**Position**
prüfen, ob ein Ding herausgenommen werden kann	
ein Ding herausnehmen	
prüfen, ob ein best. Ding enthalten ist	
ein best. Ding ermitteln	
ein Ding an einer best. Position ermitteln	
das Inhaltsverzeichnis ermitteln	
das Positionenverzeichnis ermitteln	
prüfen, ob ein Ding zum Inventar gehört	
ein Ding dem Inventar hinzufügen	
ein Ding aus dem Inventar entfernen	
die Inventarliste ermitteln	
die Raumbeschreibung ermitteln	

Abbildung 13: CRC-Karte "Raum"

Die Räume des Buchverlagsbeispiels modelliere ich nach dem Entwurf der folgenden drei Fachwerte, die noch unmittelbar zum Raum gehören.

3.2.3.3 Raumbeschreibung

Die Raumbeschreibung ist ein komplexer Fachwert. Sie besteht aus der Bezeichnung des Raumes (z.B. dem Namen oder einer Raumnummer), seiner Farbe und dem Aussehen des "Teppichbodens".

Der Sinn der Farbe und des "Teppichbodens" wird erst im Vorgriff auf das im folgenden Unterkapitel entworfene Benutzungsmodell deutlich, in dem ein Raum durch eine Art Desktop-Werkzeug visualisiert wird[284]. Farbe und "Teppichboden" dienen dabei nicht nur ästhetischen Zwecken. Sie sind zudem die augenfälligsten Merkmale eines Raumes. Daher liegt es nahe, damit auf besondere Eigenschaften eines Raumes hinzuweisen oder – bei vielen gleichartigen Räumen – künstlich Unterscheidungsmerkmale zu erzeugen. So können z.B. die

[284] Kap.3.3.1, Abb.15, S.90

persönlichen Räume von Benutzern oder alle Räume einer best. Abteilung charakteristische Farben haben. Es kann z.B. auch eine Reihe von Arbeitsräumen, die für bestimmte, zeitlich begrenzt laufende Projekte benutzt werden (z.B. in einem Architekturbüro), zur Unterscheidung bunt eingefärbt werden: "Den roten Projektraum belegt Meyer seit gestern mit dem Studentenwohnheim-Umbau. Der blaue wird morgen frei.".

Die Lage des Raumes im Gebäude ist nicht dem Raum inhärent, sondern eine Frage der Topologie des Gebäudes. Somit werden diese Merkmale nicht in der Raumbeschreibung, sondern im Gebäudeplan modelliert.

3.2.3.4 Inhalts- und Positionenverzeichnis

Inhaltsverzeichnis und Positionenverzeichnis sind miteinander verwandte, listenartige Fachwerte.
Inhaltsverzeichnisse werden in WAM als Merkmal fachlicher Behälter angesehen[285], somit auch von Räumen. Ein Inhaltsverzeichnis repräsentiert den Inhalt eines Behälters, indem es die enthaltenen Dinge aufeinanderfolgend beschreibt[286]. Die Struktur von Behälter und Inhaltsverzeichnis stimmen überein. Das Inhaltsverzeichnis ist jedoch nur eine Beschreibung mithilfe von Fachwerten. Es besitzt Wertsemantik[287].

Das Positionenverzeichnis erweitert das Inhaltsverzeichnis, indem es zusätzlich zu den Beschreibungen der Dinge auch deren Positionen auf der "Fußbodenfläche" des Raumes enthält.
In einem Raum hat jedes Ding seine Position. Insofern ist das Positionenverzeichnis gegenüber dem Inhaltsverzeichnis die angemessenere Beschreibung des Inhalts eines Raumes. Aus diesem Grund führe ich es ein, obwohl das Inhaltsverzeichnis dadurch redundant wird.

Inhaltsverzeichnis und Positionenverzeichnis führen sämtliche Dinge, die in einem Raum enthalten sind, auf – auch die inventarisierten Dinge, obwohl diese zudem in der Inventarliste erscheinen.

3.2.3.5 Inventarliste

Konzept	Zweck
stabile Topologie innerhalb einer starren Topologie (Inventar)[288]	um den Zugriffspunkt auf gemeinsame Ressourcen festlegen zu können

Die Inventarliste ist ebenfalls ein Fachwert. Sie repräsentiert das Inventar des Raumes.

---

[285] [WAM98], S.394
[286] Dazu verwendet es deren Dingbeschreibungen (Kap.1.4.3, S.51). Der Zusammenhang zwischen einem Ding im Behälter und dessen Dingbeschreibung im Inhaltsverzeichnis wird technisch über den Identifikator des Dinges (Kap.4.1.1, S.122) hergestellt, der an beiden ermittelt werden kann.
[287] Kap.1.1.3, Tabelle 2, S.22
[288] Kap.2.2.2, Tabelle 4, S.65

In der Form entspricht sie dem Inhaltsverzeichnis des Raumes, enthält aber nur die im Raum enthaltenen und inventarisierten Dinge – also eine Untermenge der Einträge des Inhaltsverzeichnisses. Betrachtet man das Inventar als einen Behälter (die entsprechenden Umgangsformen sind als Teil der Umgangsformen mit dem Raum erkennbar), so ist die Inventarliste dessen Inhaltsverzeichnis.

Zum Beispiel aus dem Buchverlag:

> Die drei zu modellierenden Räume sind: der Raum der Abteilungsleiterin, das Sekretariat und der Waschzettelraum.
> Die Benennungen wähle ich etwas kürzer – so als wären es z.B. Aufschriften auf den Türen: "Abteilungsleiterin", "Sekretariat" und "Waschzettel".
> Um den persönlichen Arbeitsraum[289] der Abteilungsleiterin von den öffentlichen Räumen optisch abzugrenzen, erhält er eine andere Farbe als diese.
> Im Sekretariat gehört die Journalistenkartei zur festen Ausstattung, im Waschzettelraum das Waschzettelarchiv. Diese Dinge befinden sich im Inventar des jeweiligen Raumes.
> Die übrige Ausstattung aller Räume ist situationsgebunden und variabel – ein Beispiel für eine mögliche Ausstattung findet sich im nachfolgenden Unterkapitel zum Benutzungsmodell[290].
>
> Die Anordnung der Räume modelliere ich im Anschluss an den Entwurf des Gebäudeplans.

3.2.3.6 Gebäude

Räume sind ohne ein Gebäude nicht vorstellbar. "Freistehende" Räume, die nicht Teil eines Gebäudes sind, existieren in der physischen Welt nicht. In diesem Sinne muss ein Gebäude auch in der virtuellen Welt vorgesehen werden, das die einzelnen Räume zu einem fachlichen Ganzen zusammenfasst. Aus diesem Grund modelliere ich die durch die Räume aufgeteilte gemeinsame Umgebung hier in Form eines virtuellen Gebäudes.

Konzept	Zweck
gemeinsame Umgebung[291]	für den Materialaustausch

Die PR besitzt in der physischen Welt kein eigenes Gebäude, sondern belegt nur einige Räume des Buchverlagsgebäudes. Dennoch ist es sinnvoll, die Ansammlung der virtuellen Räume der PR ein Gebäude zu nennen. In der virtuellen Welt hat die PR in diesem Fall ein eigenes virtuelles Gebäude[292].

---

[289] hier nicht zu verwechseln mit dem privaten Einzelarbeitsplatz
[290] Kap.3.3.1, Abb.15, S.90
[291] Kap.2.2.1, Tabelle 3, S.64
[292] ... sofern nur die PR im System modelliert wird. Soll dagegen das gesamte Buchverlagsgebäude mit seinen unterschiedlichen Abteilungen modelliert werden, so bietet es sich an, die Räumlichkeiten in Etagen und Trakten zu strukturieren (Kap.3.4.4, S.112).

Das Gebäude vergegenständlicht die gemeinsame Umgebung. Es bildet den Abschluss des fachlichen Modells der gemeinsamen Umgebung. Das bedeutet, dass alle Dinge, die sich innerhalb der gemeinsamen Umgebung befinden, in dem Gebäude enthalten sind.

Das Gebäude besteht aus Räumen, die einer starren (Nachbarschafts-)Topologie gemäß zusammengesetzt sind. Diese starre Topologie beschreibt der Gebäudeplan, welcher am Gebäude sondiert werden kann.
Das Gebäude, also die gemeinsame Umgebung, ist vollständig in Räume aufgeteilt. D.h. alle Dinge, die im Gebäude enthalten sind, sind in genau einem Raum enthalten – es gibt keine Dinge oder Plätze außerhalb von Räumen.

Das Gebäude enthält zwar Räume und kann daher in gewissem Sinne auch als Behälter aufgefasst werden. Allerdings gehört es nicht zum Umgang mit dem Gebäude, Räume einzufügen und zu entfernen. Dies ist auch nicht erwünscht, da die Topologie dann nicht mehr starr wäre. Die Orientierung innerhalb des Gebäudes würde damit erschwert. Lediglich bei der Konstruktion werden Räume quasi zum Gebäude hinzugefügt, danach ist die Struktur des Gebäudes unveränderlich[293].
Insofern fehlen die für einen Behälter charakteristischen Umgangsformen. Das Gebäude ist nur auf einer abstrakten Ebene ein Behälter, nicht auf der konkreten Ebene des Umgangs.

Alle Räume in dem Gebäude sind gleichermaßen unmittelbar erreichbar, d.h. fokussierbar. Sie sind nicht durch Türen verbunden. Wege durch das Gebäude spielen somit bei der Navigation keine Rolle.
Das menschliche räumliche Verständnis einer Umgebung ist in drei Stufen aufgebaut: "landmark, route, and survey knowledge"[294]. Durch die Modellierung von Türen würde das räumliche Verständnis des Gebäudes auf der niedrigeren Stufe "route" verbleiben müssen: Die Kenntnis genauer Wege wäre die Vorraussetzung, um z.B. von einem Raum aus einen anderen zu erreichen. Durch die direkte Erreichbarkeit der Räume, kann das Verständnis auf der höhreren Stufe des "survey knowledge" unmittelbar zur Navigation verwendet werden.

Da das Fokussieren eines Raumes eine Werkzeugeinstellung ist[295], gehört das Fokussieren nicht zum Umgang mit den Räumen oder dem Gebäude.

Gebäude	
den Gebäudeplan ermitteln	**Gebäudeplan**
prüfen, ob ein best. Raum existiert	**Raum**
einen best. Raum ermitteln	

Abbildung 14: CRC-Karte "Gebäude"

[293] Die Konstruktion eines Gebäudes sehe ich jedoch wie die Konstruktion von Materialien im Aufgabenbereich der Softwareentwickler.
[294] [Kuhn o.J.], S.4
[295] Kap.3.1.1, S.71

Das Gebäude zum Buchverlag modelliere ich im Anschluss an den Entwurf des Gebäudeplans.

3.2.3.7 Gebäudeplan

Konzept	Zweck
starre Topologie[296]	zur Orientierung in der gemeinsamen Arbeitsumgebung

Der Fachwert Gebäudeplan beschreibt die starre Topologie eines Gebäudes. Betrachtet man ein Gebäude auf einer abstrakten Ebene als einen Behälter von Räumen, so ist der Gebäudeplan dessen Inhaltsverzeichnis. Darüberhinaus enthält der Gebäudeplan jeweils den Grundriss eines Raumes, dessen Lage im Gebäude und seine Raumbeschreibung.

Das Inventar und der übrige Inhalt der Räume sind nicht im Gebäudeplan aufgeführt.

Entsprechend visualisiert, kann der Gebäudeplan zur Navigation dienen[297]. Er vermittelt einen Überblick über das Gebäude und unterstützt so ein räumliches Verständnis auf der Stufe des "survey knowledge".

Zum Beispiel aus dem Buchverlag:

> Die drei Räume werden im virtuellen Gebäude genau wie im physischen Gebäude angeordnet. Das ist naheliegend und bei nur drei Räumen zudem kein entscheidender Punkt.
>
> Die Gestaltung eines komplexeren Gebäudegrundrisses stellt in der Praxis aber sicherlich eine interessante Aufgabe dar, zu deren Lösung die Entscheidungskriterien erst noch erarbeitet werden müssen. Die virtuelle Umgebung wird vermutlich selten der physischen Umgebung in der Zusammenstellung der Räume und der Topologie so genau entsprechen. Neue Räume werden beim Übergang von der physischen in die virtuelle Umgebung hinzukommen, andere werden zusammengefasst oder aufgeteilt werden.

[296] Kap.2.2.2, Tabelle 4, S.65
[297] als Vision dazu, siehe Kap.3.4.6, S.114

3.3 Das Benutzungsmodell

Bisher hat sich noch keine Darstellungsform zur Beschreibung des Benutzungsmodells etabliert. Daher habe ich mich entschlossen, das Benutzungsmodell in Form einer Vorschau auf die schließlich realisierte Raumkomponente zu beschreiben. Hier nehme ich zwar das Ergebnis der Arbeit vorweg, jedoch scheint mir dieser Weg am anschaulichsten zu sein. In der logischen Abfolge der Arbeit mag das Unterkapitel zum Benutzungsmodell zunächst als Systemvision aufgefasst werden, dem ein bestimmtes Benutzungsmodell zugrunde liegt. Eigentlich handelt es sich aber um ein Verwendungsbeispiel und eine Beschreibung der tatsächlich realisierten Raumkomponente. Es wäre wenig sinnvoll gewesen, dieselben Punkte zweimal abzuhandeln.

Das Benutzungsmodell gliedert sich in verschiedene Aspekte, die ich jeweils in einem eigenen Unterkapitel beschreibe. Zu Beginn eines solchen Abschnitts steht ein Screenshot der realisierten Raumkomponente, in der der jeweilige Aspekt illustriert wird.
Konzepte aus dem zweiten Kapitel, die nicht schon im fachlichen Modell, sondern erst im Benutzungsmodell umgesetzt werden können, zeige ich wieder in Form des entsprechenden Tabellenausschnitts aus Kapitel 2.

Das Beispiel aus dem Buchverlag greife ich soweit sinnvoll zwischen den Abschnitten auf. Dort beschreibe ich, wie die Basis-Raumkomponente in Bezug zum jeweiligen zuvor beschriebenen Aspekt des Benutzungsmodells fachlich angepasst und verwendet werden kann. Die Szenarios spiele ich nicht im einzelnen durch – dies kann man sich leicht selbst überlegen.

3.3.1 Visualisierung der Plätze

Abbildung 15: Visualisierung der Plätze

Die obige Abb.15 gibt ein Beispiel für die Visualierung des Einzelarbeitsplatzes (links) und der Räume im gemeinsamen Gebäude (rechts). Der Einzelarbeitsplatz wurde in Form einer Desktopkomponente bereits – wie schon erwähnt – im Rahmen der Arbeit von Lippert für das JWAM-Framework realisiert[298].

Die Desktopmetapher hat sich in vielen Benutzungsoberflächen bewährt[299]. Entscheidenden Einfluss hat dabei die Tatsache, dass durch die Umsetzung der Desktopmetapher Objekte und Aktionen weitgehend aufgrund ihr Visualisierung verstanden werden können[300].
Aufgrund der Ähnlichkeit der Problematik – nämlich der Handhabung einer virtuellen Arbeitsfläche bzw. "Fußbodenfläche" – erscheint mir der Desktop als Vorbild für die Präsentation der Räume geeignet. Daher fälle ich die Entwurfsentscheidung, die gemeinsamen Räumlichkeiten desktop-artig zu präsentieren. Dies wurde bereits in [Lippert99] vorgeschlagen. Jeder Raum wird durch ein eigenes "Desktop-Werkzeug"[301] dargestellt und kann darüber auch manipuliert werden – dazu in den folgenden Unterkapiteln mehr.

Ein solches desktop-ähnliches Werkzeug zur Handhabung eines einzelnen Raumes besteht aus den folgenden visuellen Komponenten: einer Titelzeile (B), einem Rahmen (D) und einem

[298] siehe Packages de.jwamalpha.handling.desktop (Desktop-Werkzeug) und de.jwam.handling.environment (Umgebung / fachliches Modell)
[299] [Lippert99], S.23
[300] [Lippert99], S.24 (Zitat David C. Smith)
[301] [Lippert99], S.97

Bereich, der eine Draufsicht auf die "Fußbodenfläche" des Raumes visualisiert (E). Unterhalb der Titelzeile befindet sich ein kleiner Bereich (C), in dem als Awarenessinformation die "Besucher" des Raumes, d.h. die Benutzer, die den Raum fokussieren, angezeigt werden[302]. Dinge werden in Form von beschrifteten Icons repräsentiert und ihrer Anordnung auf der "Fußbodenfläche" gemäß in dem entsprechenden Bereich des Werkzeugs dargestellt.

Um die starre Topologie des Gebäudes zu visualisieren, werden die einzelnen desktop-ähnlichen Raum-Werkzeuge dem Gebäudeplan gemäß in einem gemeinsamen GUI-Frame (A) angeordnet. Dieses Werkzeug nenne ich *RoomExplorer*[303] – dies ist gleichzeitig der Name für das gesamte Gruppenwerkzeug.
Das desktop-ähnliche Werkzeug, welches einen einzelnen Raum präsentiert, ist ein Subwerkzeug des RoomExplorers. Ich nenne es *RoomTop* in Anlehnung an den Desktop, der einen Schreibtisch präsentiert.

So wie die "Fußbodenflächen" der Räume wird auch der Grundriss des Gebäudes als Draufsicht visualisiert. Der RoomExplorer-Frame ist in der Größe veränderbar, dementsprechend passen sich die eingebetteten RoomTops in ihrer Größe an[304].

Durch die Darstellung der privaten und der gemeinsamen Umgebung in getrennten Frames sind beide leicht auseinanderzuhalten. Die nur im Frame der gemeinsamen Umgebung erscheinenden Awarenessinformationen machen den Unterschied zwischen beiden Umgebungen zudem noch bewusster: offenbar arbeiten nur in der gemeinsamen Umgebung noch weitere Benutzer. Die fachliche Transparenz[305] der möglichen technischen Verteilung der Umgebungen wird dadurch gefördert.

Doch auch die Verwendung der Frames dürfte die fachliche Transparenz erhöhen.
Objekte, mit denen wir am häufigsten arbeiten, sind meist in unserer Nähe angeordnet, andere Materialien werden weiter entfernt aufbewahrt. Zudem handeln wir i.d.R. dort, wo wir sind[306].
Für die eigentliche Arbeit spielt der Frame der privaten Umgebung die größere Rolle. Dort werden benötigte Dinge bereitgelegt und Werkzeuge gestartet. Insofern ist es naheliegend, die private Umgebung hier als "nah" und die gemeinsame Umgebung als "entfernt" aufzufassen.

Das Icon zum Aktivieren des RoomExplorers befindet sich auf dem Desktop, der den Einzelarbeitsplatz präsentiert, also in der privaten Umgebung. Hier wird nicht verschleiert, dass es sich beim RoomExplorer lediglich um ein Werkzeug zur Handhabung des virtuellen Gebäudes handelt und nicht um das Gebäude selbst.

---

[302] Kap.3.3.5, Abb.20, S. 98 – als Beispiel für die Visualisierung dieser Awarenessinformationen
[303] Der Name *BuildingExplorer* erschien mir für dieses Werkzeug nicht so geeignet. Obwohl es ja ein Gebäude visualisiert, sind es aber eher die Räume, auf denen das Interesse des Benutzers liegt, und die mithilfe des RoomExplorer-Werkzeugs durchstöbert werden sollen.
[304] Kap.3.3.3, Abb.17, S. 95 – als Beispiel für einen verkleinerten RoomExplorer-Frame
[305] Kap.3.1.3, S.74
[306] [HarrisonDourish96], S.67 und S.68 (proximity and action)

Durch die desktop-ähnliche Präsentation im RoomExplorer entsteht der Eindruck, die entfernten Plätze direkt (ohne Werkzeug) zu manipulieren. Die Visualisierung des Gebäudes wirkt, als *sei* sie das entfernte Gebäude. Das mag sich auf die fachliche Transparanz evtl. wiederum nachteilig auswirken, da die technische räumliche Entfernung des gemeinsamen Gebäudes u.U. vom Benutzer schlicht vergessen wird. Der entscheidende Vorteil der desktop-ähnlichen Präsentation liegt jedoch in der geringen visuellen Vorstellungskraft, die notwendig ist, damit sich der Benutzer ein Bild von der gemeinsamen Umgebung machen kann. Dieses Bild wird ihm durch die desktop-ähnliche Präsentation unmissverständlich vor Augen geführt – und ist sogar direkt manipulierbar.

Konzept	Zweck
Einheit von Raum und Zeit[307]	um eine räumliche Vorstellung von Plätzen zu ermöglichen

Das Konzept der Einheit von Raum und Zeit, welches dem gesamten Benutzungsmodell zugrundeliegt, wird in der Präsentation sichtbar. Sämtliche Dinge, die sich in der individuellen Umgebung eines Benutzers befinden, werden in genau einem der RoomTops oder auf dem Desktop visualisiert.

Alle Dinge, die sich in einem Raum befinden, sind an ihrer jeweiligen Position im RoomTop sichtbar – es ist nicht möglich, Dinge auszublenden[308] und so verschiedenen Benutzern unterschiedliche Einblicke in den Raum zu gewähren[309].

Konzept	Zweck
alle Dinge an einem gemeinsamen Platz sind unmittelbar und für jeden an diesem Platz gleichermaßen sichtbar...	um sich über den gemeinsamen Platz verständigen zu können
... und zugreifbar[310]	um den Handlungsrahmen eines Kooperationspartners einschätzen zu können

Tatsächlich stellen sich die gemeinsamen Räumlichkeiten allen Benutzern auf weitgehend dieselbe Weise dar. Man spricht in diesem Zusammenhang auch von WYSIWIS – "What You See Is What I See"[311]. In Bezug auf den Zustand der Räume und ihres Inhalts besteht eine strenge WYSIWIS-Kopplung[312] zwischen den RoomExplorern der verschiedenen Benutzer. So wird z.B. die Anordnung der Räume und Dinge überall auf die gleiche Weise dargestellt.
Auf diese Weise können sich die Kooperationspartner über die gemeinsame Umgebung verständigen ("Welche Notiz gehört zum Mailing?" – "Die, die direkt darunter liegt." (siehe Abb.15)).

[307] Kap.2.2.1, Tabelle 3, S.64
[308] analog zum Datei-Attribut "hidden"
[309] sollen Dinge nicht sichtbar sein, so können sie stattdessen z.B. in einem Behälter "versteckt" werden – analog zur physischen Welt
[310] Kap.2.2.1, Tabelle 3, S.64 (beide Konzepte)
[311] [Mayer94], S.4
[312] Strenge WYSIWIS-Kopplung bedeutet, dass sich an den Clients die Visualisierungen eines Systemausschnitts genau gleichen. ([Mayer94], S.16)

Die Darstellungen enthalten aber auch private Anteile (z.B. die Scrollposition in einem RoomTop) – dies werden wir im Folgenden noch sehen.
Darüberhinaus können alle Benutzer mit ihrem RoomExplorer das gleiche tun und in gleicher Weise auf die Räume und die darin enthaltenen Ressourcen zugreifen[313] – es gilt also in Analogie zum WYSIWIS: "What You Can Do Is What I Can Do".

Zum Beispiel aus dem Buchverlag:

Die Ausstattung der Einzelarbeitsplätze besteht aus den folgenden Dingen:

- einer Werkzeugkiste (Toolbox), in der sich z.B. Werkzeuge zum Schreiben des Mailings oder zum Durchstöbern der Journalistenkartei befinden,
- einer Mappe mit Materialvorlagen (Material Templates), wie z.B. Notizzettel, Mailingbögen, leeren Anfragenlisten usw.,
- einem Papierkorb (Trash Can),
- dem RoomExplorer und
- jeweils einem Postein- und -ausgangskorb (In- und Outbox), um mit den Journalisten oder auch untereinander explizit durch Materialaustausch zu kommunizieren und zu kooperieren.

Alle Einzelarbeitsplätze der Mitarbeiterinnen der PR sind mit den gleichen Dingen ausgestattet, da alle Mitarbeiterinnen prinzipiell für die gleichen Aufgaben zuständig sind.

Die Aufteilung der gemeinsamen Umgebung in die drei Räume der PR-Abteilung lässt sich in der Abb.15 gut erkennen – offenbar ist kaum visuelle Phantasie vonnöten, um sich ein Bild vom Aufbau der gemeinsamen Umgebung zu machen.

Um für die einzelnen Benutzerinnen einen Eindruck von konventionsgemäß eher persönlichen Bereichen der gemeinsamen Umgebung entstehen zu lassen, werden die Räume farblich unterschieden. Das ist eine informale Gestaltungsmöglichkeit.
Der Raum der Abteilungsleiterin wird – als ihr persönlicher Raum – von den übrigen Räumen durch einen violettfarbenen "Teppichboden" optisch abgesetzt. Dies ist zudem die Farbe des Desktops der Abteilungsleiterin. Die Desktops der Mitarbeiterinnen des Sekretariats (in der Abb.15 nicht zu sehen) haben – wie die öffentlichen Räume – einen blauen Hintergrund.

Die Visualisierung in Abb.15 orientiert sich an den Kooperationsbildern des zweiten Kapitels[314].
Sowohl der RoomExplorer als auch die Kooperationsbilder visualisieren Kooperationssituationen innerhalb einer gemeinsamen Umgebung. Die visuellen Elemente des RoomExplorers ähneln denen, die in den Kooperationsbildern zum

[313] Abschließbare Räume (Kap.3.4.3) wären da keine Ausnahme, da die Zugriffsmöglichkeit lediglich eine Frage des richtigen Schlüssels ist und keine Frage des RoomExplorers.
[314] Kap.2.1.1, Abb.8, S.59 und Kap.2.1.2, Abb.9, S.60

Einsatz kommen (z.B. Rechtecke für Räume, icon-ähnliche Symbole für Dinge usw.).
Die Struktur der Kooperationsbilder läßt sich auf diese Weise gut auf den RoomExplorer übertragen – obwohl die Visualisierung des RoomExplorer und die Kooperationsbilder unterschiedlichen Zwecken dienen und in der einen Darstellungsform virtuelle, in der anderen physische Plätze dargestellt werden.
Ist eine Kooperationssituation mithilfe von Kooperationsbildern dokumentiert worden, so lässt sie sich in einem geeignet entworfenen RoomExplorer ohne gravierenden Bruch in der Darstellung nachvollziehen.
Zudem ist durch die ähnliche Darstellungsweise für Entwickler und Benutzer offensichtlich, inwieweit der RoomExplorer die räumliche Situation aus den Kooperationsbildern wiedergibt und an welchen Stellen er ggf. angepasst werden muss.

3.3.2 Fokussieren eines Raumes

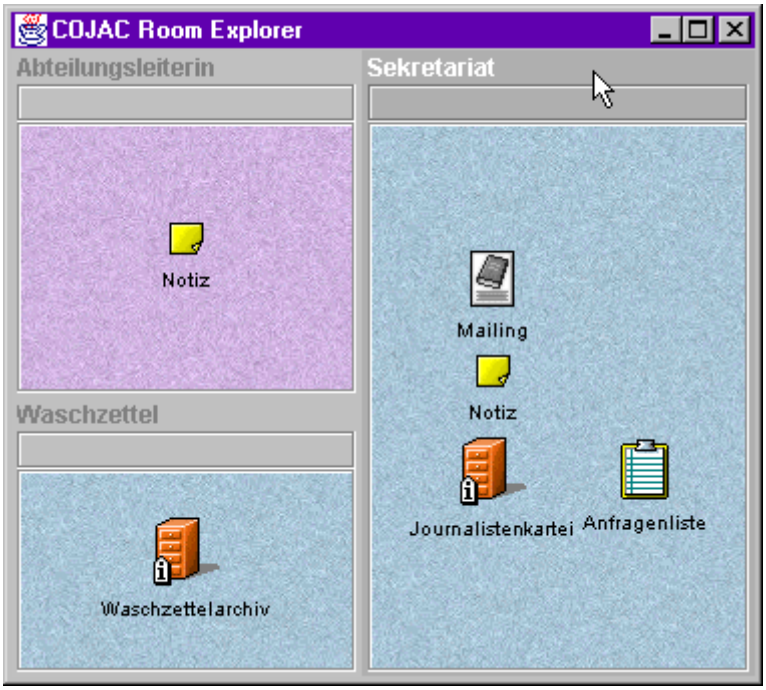


Abbildung 16: Fokussieren eines Raumes

Virtuelle Räume werden, wie bereits dargelegt, nicht betreten, sondern mithilfe eines Werkzeugs fokussiert[315]. Dies geschieht hier durch das Setzen des Tastaturfokus auf einen RoomTop, der dem gewünschten Raum entspricht. Den Tastaturfokus setzt man per Mausklick auf einen der Bereiche des RoomTops, z.B. auf die Titelzeile, die "Fußbodenfläche" oder auf eines der enthaltenen Dinge. Alle Tätigkeiten am RoomExplorer lassen sich grundsätzlich mit der Maus vollziehen, wie in den folgenden Unterkapitel zu sehen sein wird.

[315] Kap.3.1.1, S.71

Rahmen und Titelzeile des RoomTops des fokussierten Raumes werden farblich abgehoben. Es kann immer nur ein einzelner Raum zu einem Zeitpunkt fokussiert werden. Da das Konzept des Fokussierens von virtuellen Räumen dem Betreten von physischen Räumen entspricht, ist es nur konsequent, die Einheit von Raum und Zeit auch auf den Fokus zu beziehen: So, wie der Benutzer in der physischen Welt zu einem best. Zeitpunkt nur in genau einem Raum befinden kann, kann er in der virtuellen Welt zu einem best. Zeitpunkt auch nur einen Raum fokussieren.
Diese Einheit von Raum und Zeit, der der Fokus unterliegt, kann gut über den Tastaturfokus umgesetzt werden, da sich dieser zu einem Zeitpunkt immer nur auf ein einzelnes GUI-Widget bezieht: Klickt der Benutzer den Desktop-Frame an, so verliert der RoomExplorer-Frame und damit auch das zuvor angeklickte RoomTop-Widget seinen Fokus – der entsprechende Raum wird vom Benutzer nicht mehr fokussiert[316].

3.3.3 Ansichten eines Raumes – Scrollen und "Zoomen"

Abbildung 17: Scrollen

Wird der Bereich des RoomTops, der die "Fußbodenfläche präsentiert, zu klein, um alle darauf angeordneten Dinge anzuzeigen, werden nach Bedarf vertikale oder horizontale Scrollbars eingeblendet. Die Funktionalität ist allgemein bekannt.
Aufgrund der unbegrenzten Kapazität eines Raumes ist es notwendig, in einem RoomTop scrollen zu können. Die "Fußbodenfläche" entspricht somit allerdings nicht der Grundrissfläche des Raumes im RoomExplorer, da sie auch größer sein kann als diese.

[316] in der Implementation zu dieser Arbeit noch nicht realisiert – dort behält der zuvor fokussierte RoomTop seinen farblich abgesetzten Rahmen auch dann, wenn der Desktop den Fokus erhält

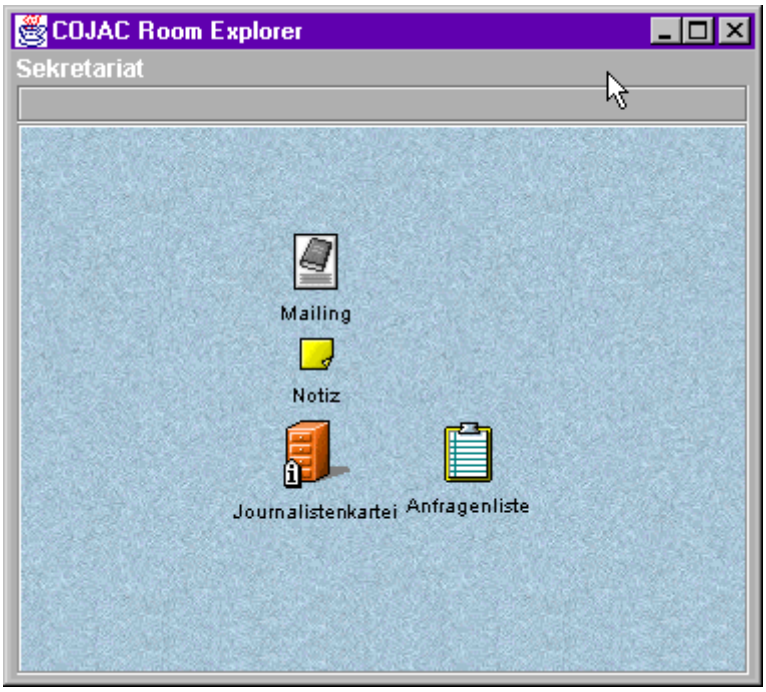


Abbildung 18: "Zoomen" in Raumansicht

Es sind zwei verschiedene Ansichten auf die visualisierten Räume einstellbar: In der *Raumansicht* wird der fokussierte Raum einzeln dargestellt (siehe Abb.18) und in der *Gebäudeansicht* wird eine Gesamtansicht auf alle Räume des Gebäudes gezeigt (siehe Abb.15).

Durch einen Doppelklick auf die Titelzeile eines RoomTops kann dieser auf die volle Größe des Client-Bereichs[317] des RoomExplorer-Frames vergrößert werden – der Raum wird quasi "herangezoomt". Dabei wird er ggf. gleichzeitig fokussiert. Durch einen weiteren Doppelklick auf die Titelzeile des RoomTops kann von der Raumansicht wieder in die Gebäudeansicht zurückgewechselt werden[318].

Die Raumansicht ist sinnvoll, um z.B. aus mehreren Projekträumen den mit dem aktuell zu berarbeitenden Projekt "heranzuzoomen" – zum einen steht dadurch mehr Bildschirmfläche für den Raum zur Verfügung, zum anderen können evtl. störende Awarenessinfos aus anderen Räumen auf diese Weise ausgeblendet werden.

Die Gebäudeansicht ist z.B. sinnvoll, um Dinge von einem Raum in den anderen zu bewegen[319] oder um einen anderen Raum in die Raumansicht zu schalten.

Andere Benutzer können nicht erkennen, ob die Raum- oder Gebäudeansicht gewählt wurde oder welcher Ausschnitt des Raumes sich im sichtbaren Scrollbereich befindet. Hier wird die strenge WYSIWIS-Kopplung der RoomExplorer abgeschwächt.

---

[317] Das ist die Fläche des gesamten Frames abzüglich des Rahmens und der Titelzeile – also der Bereich, in dem die enthaltenen Widgets des Frames (seine Clients) angezeigt werden.
[318] Diese Funktionalität ist z.B. bereits aus Win95 bekannt. Dort kann durch einen Doppelklick auf die Titelzeile eines Fensters zwischen Vollbild- und Normalansicht umgeschaltet werden.
[319] Kap. 3.3.6, S.101

Zum Beispiel aus dem Buchverlag:

Da die Mitarbeiterinnen des Sekretariats in erster Linie auf die Ressourcen des Sekretariats zugreifen, bietet es sich für sie an, das Sekretariat dauerhaft in Raumansicht "heranzuzoomen". So haben sie eine bessere Übersicht über die enthaltenen Dinge (die mit der Zeit sicher zahlreicher sein werden, als in Abb.18) und müssen z.B. weniger oft scrollen.

3.3.4 Reservieren einer Ressource durch Markieren

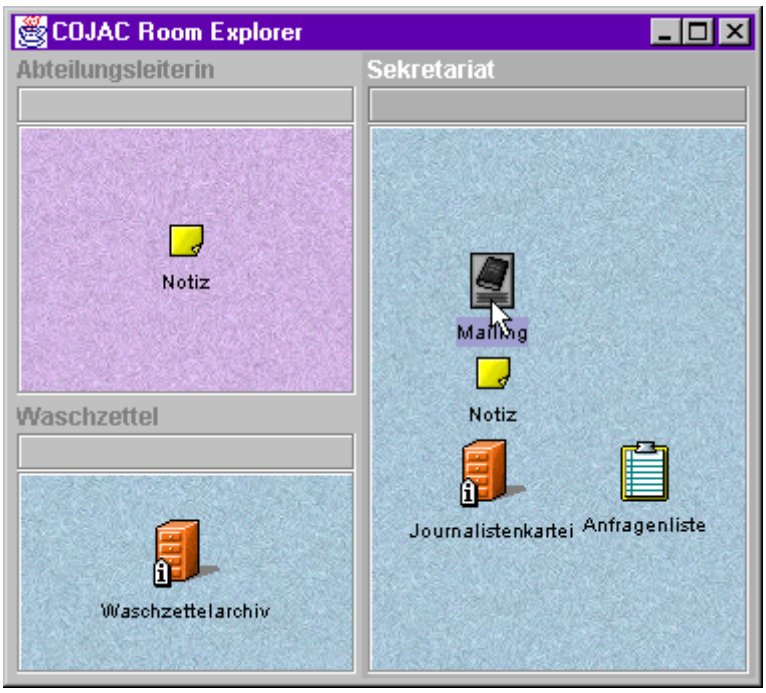


Abbildung 19: Reservieren durch Markieren

Markiert ein Benutzer eine Ressource, so werden ihr Icon und die Beschriftung dunkel unterlegt.

Durch das Markieren reserviert sich der Benutzer die Ressource für die exklusive Nutzung. Er kann die Ressource nun bewegen (z.B. an seinen Einzelarbeitsplatz[320]) oder auf andere Weise direkt manipulieren (z.B. über ein Kontextmenü[321]). Andere Benutzer können währenddessen nicht mehr auf die Ressource zugreifen[322]. Konfliktsituationen werden also bereits beim Markieren einer Ressource bemerkt und aufgelöst. Das ist ein entscheidender Punkt, der das Benutzungsmodell maßgeblich vereinfacht.

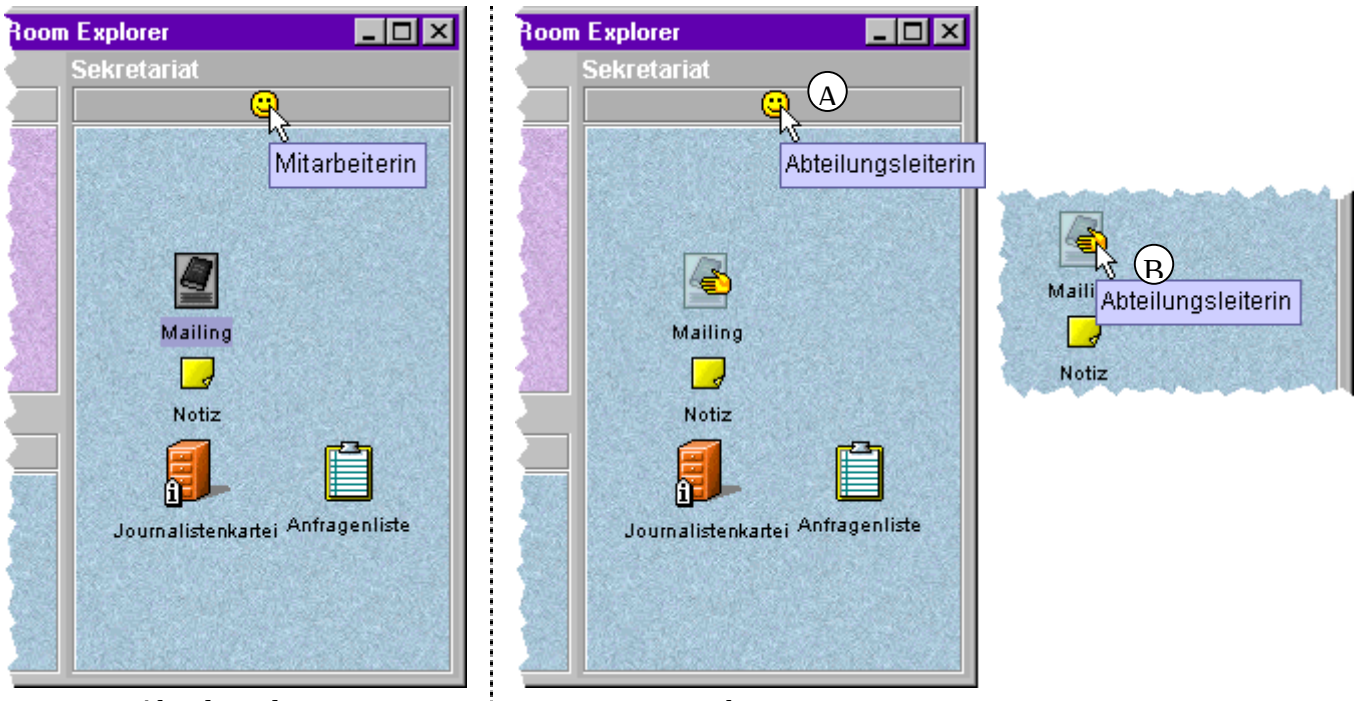
Die Ressource wird – wie beim Desktop – durch Anklicken des Icons, welches die Ressource repräsentiert, markiert. Gleichzeitig wird dabei der Raum fokussiert, in dem sich die

[320] Kap.3.3.6, Abb.21 (A), S.101
[321] Kap.3.4.1, S.107
[322] siehe folgendes Unterkapitel

Ressource befindet. Hierdurch wird nachgebildet, dass sich ein Benutzer in der physischen Welt an den Platz bewegen muss, an dem er auf eine Ressource zugreift.

Ein Benutzer kann nur eine Ressource auf einmal reservieren – das genügt zumeist. Es ist aber vermutlich eine sinnvolle Erweiterung, auch das Markieren von mehreren Ressourcen innerhalb eines Raumes zu unterstützen. Dann können auch mehrere Ressourcen auf einmal bewegt werden – z.B. an den Einzelarbeitsplatz, um sie dort exklusiv zu bearbeiten.

3.3.5 Awareness



Abteilungsleiterin Mitarbeiterin

Abbildung 20: Awareness

Die Awareness dient u.a. der Transparenz bei der impliziten Kooperation:

> "Das zentrale Problem bei der impliziten Kooperation ist demnach die Transparenz bei der Verwendung gemeinsamer Materialien. Transparenz bedeutet hier, daß der Benutzer die Konkurrenzsituation mit Hilfe des Anwendungssystems erkennen kann:
>
> - Das System muß verdeutlichen, daß mehr als ein Benutzer am gleichen Material arbeitet.
> - Die Einheit von Ort und Zeit bei der Materialbearbeitung bleibt gewahrt.
> - Die Koordination erfolgt im wesentlichen durch Konventionen außerhalb des Systems."[323]

Hier ist jedoch zu berücksichtigen, dass das Benutzungsmodell des RoomExplorers nicht unmittelbar vorsieht, dass ein Material von mehreren Benutzern gleichzeitig bearbeitet

[323] [WAM98], S. 877

werden kann. Dies wird eher durch die Benutzungsmodelle von entsprechenden Gruppenwerkzeugen umgesetzt, die das ermöglichen[324].

Stattdessen verdeutlicht der RoomExplorer, dass mehr als ein Benutzer im selben Raum arbeitet – die Awareness ist hier also weiter gefasst und beschränkt sich nicht auf tatsächliche Konkurrenzsituationen[325]. Harrison und Dourish weisen darauf hin, das Handlungen sich i.d.R. auf Objekte in der Nähe beziehen (proximity and action)[326]. Fokussieren zwei Benutzer denselben Raum, so befinden sich dieselben Objekte in ihrer "Nähe" und eine Konkurrenzsituation wird wahrscheinlich.

Konzept		Zweck
Awareness	in der gemeinsamen Umgebung sind die Handlungen der anderen unmittelbar erkennbar und...	um Konkurrenzsituationen abzusehen und differenzierter einschätzen zu können
	... können sich die Handelnden gegenseitig unmittelbar erkennen[327]	um mit dem Kooperationspartner explizit kommunizieren zu können und um Konkurrenzsituationen differenzierter einschätzen zu können

Als Awarenessinformation werden der fokussierte Raum und die markierte Ressource eines Benutzers allen anderen Benutzern angezeigt.

Harrison und Dourish nennen die drei räumlichen Aspekte *Aura*, *Fokus* und *Nimbus* als ein Mittel, um Awareness umzusetzen[328].
Die Aura steht für die Anwesenheit eines Benutzers, die hier durch den fokussierten Raum dargestellt wird. Der Fokus repräsentiert die Aufmerksamkeit des Benutzers – in diesem Benutzungmodell durch die markierte Ressource im fokussierten Raum ausgedrückt. Der Nimbus steht schließlich für den Einfluss, den der Benutzer hat und der sich hier über den gesamten Bereich seiner individuellen Umgebung erstreckt. Die Schnittmenge der Nimbi aller Benutzer ist die gemeinsame Umgebung, also das ganze Gebäude[329].
Grundlage für die Struktur der Awarenessinformationen ist die topologische Struktur des Gebäudes.

Da die gemeinsame Umgebung in allen RoomExplorern auf gleichwertige Weise präsentiert wird, werden dadurch natürlich auch die Handlungen der Benutzer – z.B. das Verschieben

[324] das Konzept der Gruppenwerkzeuge wird in Kap.3.4.1, S.105 ausführlich erläutert
[325] Der Raum ist das Material auf dem das Werkzeug RoomExplorer bzw. das Subwerkzeug RoomTop arbeitet. Insofern entspricht es der geforderten Transparenz, im RoomExplorer zu verdeutlichen, dass mehr als ein Benutzer im selben Raum arbeitet. Diese Awareness verdeutlicht jedoch nicht unmittelbar eine Konkurrenzsituation, da um die Räume der gemeinsamen Umgebung i.A. nicht konkurriert wird. Es gehört vielmehr zum Normalfall, dass mehrere Benutzer parallel darin arbeiten.
[326] Kap.1.3.5, S.39
[327] Kap.2.2.2, Tabelle 4, S.65 (beide Konzepte)
[328] [HarrisonDourish96], S.67
[329] [Rodden96] bezieht den Nimbus nicht auf Plätze, sondern auf die bearbeiteten Objekte. Danach gehören die markierten Objekte der anderen Benutzer nicht zum Nimbus eines Benutzers.

eines Materials – indirekt über ihren Effekt auf die gemeinsame Umgebung erkennbar. Dies zähle ich jedoch nicht zur Awareness, da die Handlung selbst den anderen Benutzern nicht verdeutlicht wird[330]. Dennoch dient dies der impliziten Kommunikation.
Die Handlungen selbst an allen RoomExplorern zu visualisieren, würde m.E. zu einer Flut von irrelevanten Awarenessinformationen führen. Über den fokussierten Raum können sich anbahnende Konkurrenzsituationen erkannt werden und über die markierte Ressource werden die Benutzer bei der Koordination unterstützt. Der RoomExplorer bietet damit die minimalen Voraussetzungen zur Kooperation. Weitere Arten der Awareness sehe ich hier, im Benutzungsmodell der Basis-Raumkomponente, nicht vor.

Fokussiert ein Benutzer einen Raum, so sehen die anderen Benutzer in dem entsprechenden RoomTop ihres RoomExplorers in dem Bereich unter der Titelzeile einen Smiley (A) angezeigt, welcher den Benutzer repräsentiert. Fokussieren mehrere Benutzer einen Raum, so sehen sie mehrere Smileys. In der obigen Abb.20 sehen sowohl die Abteilungsleiterin als auch die Mitarbeiterin jeweils den Smiley der anderen, da sie beide das Sekretariat fokussieren.
Markiert ein Benutzer eine Ressource, so wird das entsprechende Icon dieser Ressource bei den anderen Benutzern halbtransparent dargestellt und mit einem Handsymbol (B) versehen.

Der Name des Kooperationspartners, der den Raum fokussiert bzw. die Ressource markiert hat, erscheint, wenn die Maus über den entsprechenden Smiley (A) bzw. das Handsymbol (B) geführt wird.

Smiley und Handsymbol sind die Standardsymbole für das Fokussieren und Markieren durch andere Benutzer. Es ist sicherlich ratsam, benutzerspezifische Symbole zu verwenden, z.B. ein kleines Portraitfoto des Benutzers oder ein persönliches Icon (ähnlich den Bildchen an einer Kindergartengarderobe). Der konkrete Benutzer könnte dann am Symbol erkannt werden, ohne dass erst die Maus über das Symbol geführt werden muss.

Die hier unterstützte Awareness ist "symmetrisch", d.h. alle Benutzer nehmen sich in gleichwertiger Weise wahr[331]. Dadurch wird die Awareness dem erfahrenen Benutzer transparenter. Er weiß, was andere Benutzer innerhalb der öffentliche Umgebung von ihm wahrnehmen können, da er es seinerseits bei ihnen wahrnehmen kann.
Die vom RoomExplorers angezeigten Awarenessinformationen dienen weniger dem Beobachten oder gar dem Überwachen der anderen Benutzer, sondern vielmehr dem Erkennen von unmittelbaren Konkurrenzsituationen ("jemand schnappt dem anderen etwas direkt vor der Nase weg").

[330] Handlungen könnten z.B. visualisiert werden, indem *Telepointer* (das sind quasi die Mauszeiger der anderen Benutzer) angezeigt werden (siehe [Greenberg98], S.10 und [Mansfield o.J. 2], S.13). Eine abgeschlossene Drag&Drop-Operation (eines anderen Benutzers) könnte aber auch durch einen zeitweise eingeblendeten Pfeil von der Start- zur Zielposition dargestellt werden – ähnlich den Pfeilen in den Kooperationsbildern aus Kap.2.
[331] "Gleichwertig" bedeutet nicht "auf die gleiche Weise". Die Visualisierung des eigenen fokussierten Raumes durch eine Umrandung des RoomTops unterscheidet sich z.B. deutlich von den Smileys, mit denen der jeweilige Fokus der anderen Benutzer angezeigt wird. Auf die Weise werden Verwechslungen der Art ausgeschlossen, dass z.B. der eigene Fokus für den eines anderen Benutzers gehalten wird.

Zum Beispiel aus dem Buchverlag:

So kann es z.B. vorkommen, dass zwei Mitarbeiterinnen praktisch im selben Augenblick auf das Mailing im Sekretariat zugreifen. In diesem Fall "gewinnt" die Schnellere von beiden und markiert das Icon des Mailings. Damit hat sie es für sich reserviert. Ihr Handsymbol erscheint nun im RoomExplorer der langsameren Kolleginnen über Mailing-Icon. Diese erkennt daran, dass ihr Versuch, das Mailing zu markieren, fehlgeschlagen ist und dass sie vorerst nicht mehr auf das Mailing zugreifen kann. Benötigt sie es dennoch, so bleibt ihr, sich direkt an ihre Kollegin zu wenden, um sie zu bitten, die Ressource wieder freizugeben[332].

3.3.6 Bewegen von Dingen

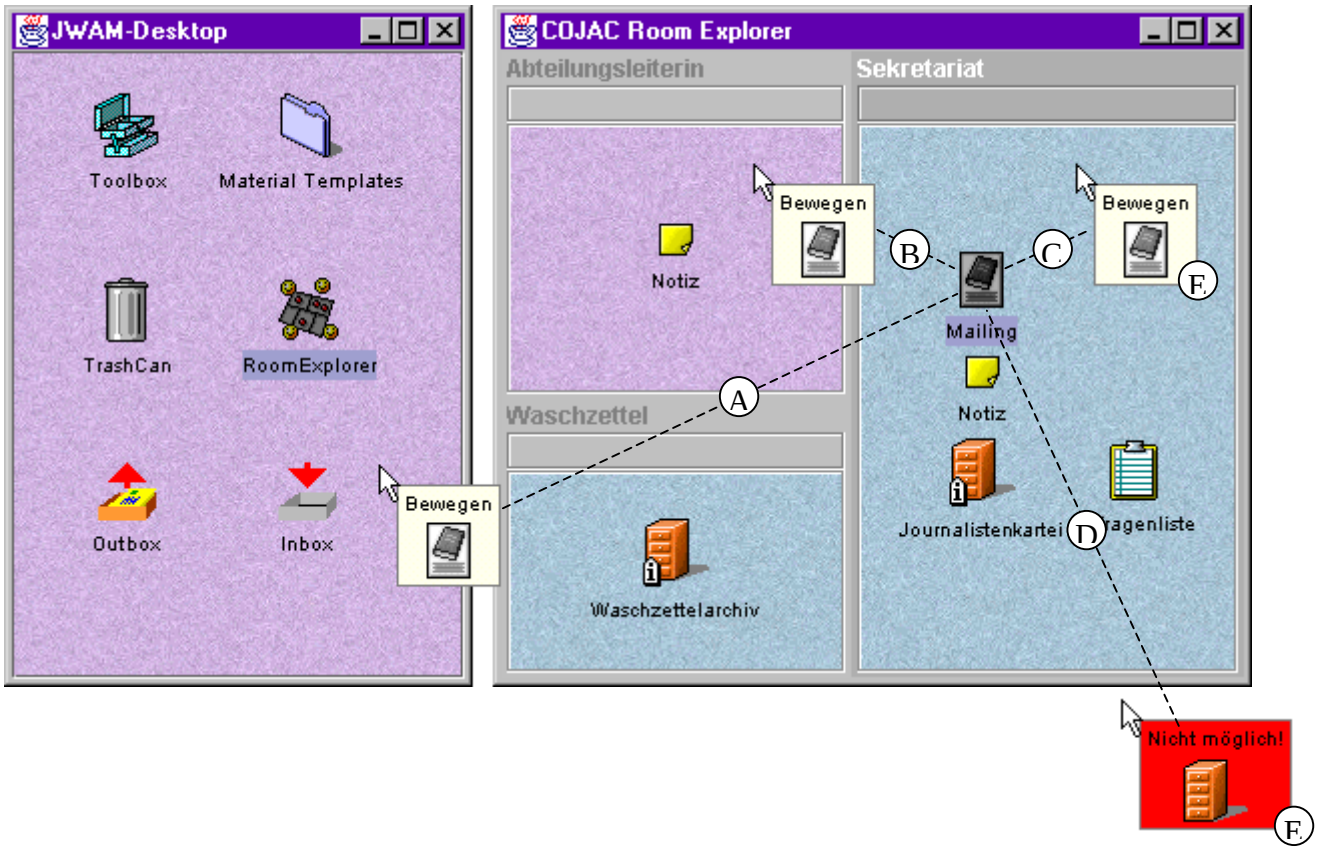


Abbildung 21: Bewegen von Dingen

Das Bewegen von Dingen geschieht beim RoomExplorer wie beim Desktop via Drag&Drop. Die Drag&Drop-Funktionalität ist allgemein bekannt.
Roock und Wolf sehen Drag&Drop als logische Fortführung des räumlichen Gedankens im Systementwurf. Dem Benutzer bietet es eine intuitive Möglichkeit, Dinge in einer virtuellen räumlichen Umgebung zu bewegen[333].

---

[332] z.B. mündlich, wenn die Mitarbeiterinnen sich im selben physischen Arbeitsraum befinden (das dürfte bei den Mitarbeiterinnen des Sekretariats der Fall sein) oder z.B. per Chat-Tool (Kap.3.4.5, S.113)
[333] [RoockWolf98], S.116

Es können nur Dinge bewegt werden, die markiert und damit für die exklusive Benutzung reserviert worden sind. Auf diese Weise ist von vornherein ausgeschlossen, dass zwei Benutzer zur selben Zeit dasselbe Ding an unterschiedliche Positionen zu bewegen versuchen.

Eine Drag&Drop-Operation beginnt mit dem Drücken der Maustaste. Sollte die Ressource, über deren Icon sich der Mauszeiger befindet, noch nicht markiert sein, wird nun versucht, die Ressource zu markieren. Das Markieren kann also implizit mit der Drag&Drop-Operation erfolgen. Schlägt der Versuch fehl (z.B. weil die Ressource bereits von einem anderen Benutzer markiert ist), wird die Drag&Drop-Operation nicht ausgeführt. Der Drag&Drop-Versuch wird ignoriert, auch wenn der Benutzer die Maus mit gedrückter Maustaste bewegt.

Die Semantik des Drag&Drop ist ausschließlich das Bewegen, nie das fachliche Kopieren von Dingen. Das dabei bewegte Ding wird an der Position an dem Platz abgelegt, über dem sich der Mauszeiger beim Loslassen der Maustaste gerade befindet. Dinge können via Drag&Drop an eine andere Position im selben Raum (C) oder an eine Position in einem anderen Raum (B) bewegt werden.

Konzept	Zweck
eine Person kann Dinge zwischen ihrer privaten Umgebung und der gemeinsamen Umgebung austauschen	um zwischen vollkommener Handlungssicherheit und der Möglichkeit zur Kooperation frei wechseln zu können

Darüberhinaus können Dinge via Drag&Drop an den Einzelarbeitsplatz bewegt werden und desgleichen zurück (A). Innerhalb der individuellen Umgebung eines Benutzers existieren also keine unsichtbaren "Barrieren", die ihn daran hindern, Dinge umherzubewegen[334]. Erwartungsgemäß ist es jedoch nicht möglich, Dinge aus der individuellen Umgebung herauszubewegen (D).

Während der Drag&Drop-Operation wird dem Benutzer angezeigt, welchen Effekt die Operation haben wird (E) – hier z.B. "Bewegen" oder keinen Effekt ("Nicht möglich!"). Dieses semantische Feedback[335] wird bereits durch die JWAM-Drag&Drop-Komponenten[336] realisiert und auch von der Desktop-Komponente verwendet.

Auf dem JWAM-Desktop können Dinge durch Drag&Drop nicht ausschließlich bewegt werden. Daneben hat Drag&Drop dort die Funktion, Dinge zueinander in Beziehung zu setzen[337]. Um ein Material zu bearbeiten, kann z.B. das Werkzeug via Drag&Drop auf das Material gezogen werden (oder umgekehrt). Diese Möglichkeit sehe ich für den RoomExplorer der Basis-Raumkomponente nicht vor.

---

[334] sichtbare "Barrieren" bilden dagegen abschließbare Räume (Kap.3.4.3, S.110) und das Inventarkonzept (Kap.3.3.7, S.103)
[335] [Lippert99], S.36
[336] siehe Package de.jwamx.technology.iafpf.dragdrop
[337] [Lippert99], S.41

Wird ein Ding an einen Platz bewegt, so wird dieser automatisch fokussiert – in Analogie zur physischen Welt: Der Benutzer trägt ein Ding an einen anderen Platz.
Die Semantik von Drag&Drop ist hier genaugenommen also nicht das Bewegen, sondern das Tragen von Dingen.

3.3.7 Inventar

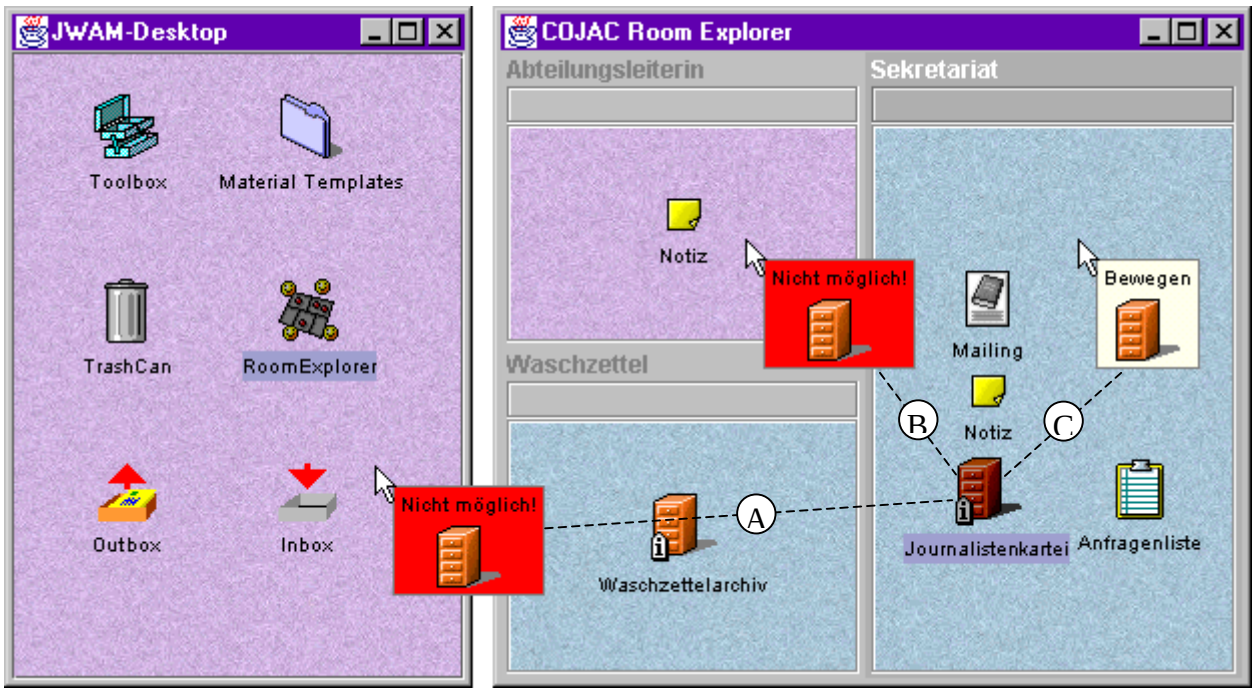


Abbildung 22: Bewegen von Inventar

Dinge, die zum Inventar eines Raumes gehören, werden im RoomExplorer mit einem stilisierten Inventar-Schildchen gekennzeichnet.

Inventarisierte Dinge sind an ihren Raum gebunden. Innerhalb ihres Raumes können sie ohne weiteres an eine andere Position verschoben werden (C). Es ist jedoch nicht möglich, inventarisierte Dinge aus ihrem Raum heraus in einen anderen zu verschieben (B). Desgleichen können inventarisierte Dinge nicht an den Einzelarbeitsplatz bewegt werden (A).

Das hat für die Bearbeitungsmöglichkeiten von inventarisierten Dingen Konsequenzen: Inventarisierte Dinge müssen in dem Raum bearbeitet werden, in dem sie sich befinden und zu dessen Inventar sie gehören. Hierzu sind Gruppenwerkzeuge[338] notwendig, wie z.B. der JWAM-Registrar[339]. Über Gruppenwerkzeuge können entfernte Ressourcen in der gemeinsamen Umgebung bearbeitet werden.

[338] Gruppenwerkzeuge werden in Kap.3.4.1, S.105 ausführlich beschrieben
[339] [JWAM00 5], S.82ff., außerdem siehe Package de.jwamx.handling.registry

Über ein spezielles Werkzeug können Dinge inventarisiert werden. Dieses Werkzeug habe ich im Rahmen des Entwurfs der Basis-Raumkomponente nicht weiter berücksichtigt[340].
Ein solches Werkzeug listet z.B. den Inhalt eines Raumes auf. Dinge, die inventarisiert oder aus dem Inventar entfernt werden sollen, können markiert und die Änderung per Knopfdruck übernommen werden.
Evtl. ließe sich eine solche Funktionalität geschickt als ein Subwerkzeug der jeweiligen RoomTops einbetten: In einem Kontextmenü, das über einem Ding-Icon per rechtem Mausklick geöffnet werden kann, kann z.B. für das repräsentierte Ding der Menüpunkt "zum Inventar hinzufügen" bzw. bei inventarisierten Dingen "aus dem Inventar entfernen" angeboten werden.

[340] Bei der Konstruktion des RoomExplorers habe ich diese Funktionalität jedoch an der Schnittstelle der Client-Komponente des RoomExplorers vorgesehen. Dort ist z.B. festgelegt, dass nur das markierte Ding eines Benutzers inventarisiert bzw. aus dem Inventar entfernt werden kann. Ein Inventarisierer-Werkzeug, das sich z.B. lokal auf dem Einzelarbeitsplatz der Abteilungsleiterin befindet, kann so leicht realisiert werden (Kap.4.2.3, Schnittstelle 14, S.140)

3.4 Die erweiterte COJAC Raumkomponente

Bei der Entwicklung der Basis-Raumkomponente sind bereits einige sinnvolle Erweiterungen aufgefallen, für die ich hier als Ausblick auf eine erweiterte Raumkomponente jeweils eine Vision entwerfe[341].

Die Erweiterungen beziehen sich auf sehr verschiedene Aspekte der Raumkomponente. Im Einzelnen handelt es sich um:

- Gruppenwerkzeuge zum Arbeiten in Räumen,
- Möbel und Stapel als in Räumen eingebettete Plätze,
- abschließbare Räume,
- Etagen / Trakte zum Gruppieren von Räumen,
- eine Chat- / Audio-Verbindung zum expliziten Kommunizieren in Räumen,
- ein Gebäudeplan-Menü zum Zugriff auf das Gebäude aus Werkzeugen heraus und
- ein Web-Interface zum Zugriff auf das Gebäude über einen Web-Server.

Bei den Gruppenwerkzeugen handelt es sich nicht direkt um eine Erweiterung der Raumkomponente selbst. Sie erweitern jedoch die begrenzten Möglichkeiten der Raumkomponente bzgl. des Zugriffs auf die gemeinsame Umgebung. Die Raumkomponente bietet einen Kontext für diese Gruppenwerkzeuge, ähnlich wie die Desktop-Komponente für die herkömmlichen "Einzel-Werkzeuge".
Eingebettete Plätze und abschließbare Räume, sowie Etagen und Trakte sind Erweiterungen des fachlichen Modells der Raumkomponente (die sich natürlich auf das Benutzungsmodell auswirken).
Die Chat- bzw. Audio-Verbindung, das Gebäudeplan-Menü und das Web-Interface erweitern unmittelbar das Benutzungsmodell.

3.4.1 Gruppenwerkzeuge

Der bereits erwähnte Registrar, mit dem auf die Dokumente in einer Registratur zugegriffen werden kann, und der RoomExplorer sind Beispiele für Gruppenwerkzeuge.

Gruppenwerkzeuge dienen der Kooperationsunterstützung. Sie ermöglichen Benutzern den Zugriff auf Ressourcen in der gemeinsamen Umgebung.
In diesem Sinne sind Gruppenwerkzeuge Groupware[342].

Im Zusammenhang mit der Basis-Raumkomponente werden sie u.a. benötigt, um inventarisierte Dinge zu bearbeiten, da diese nicht aus der gemeinsamen Umgebung in die private Umgebung eines Benutzers verschoben und somit nicht mit herkömmlichen "Einzel-Werkzeugen" bearbeitet werden können.

---

[341] bei den hier gezeigten Abbildungen handelt es sich, im Ggs. zu den Abbildungen im vorigen Kapitel, nicht um Screenshots einer bereits realisierten Komponente
[342] siehe Definition von Groupware, Kap.1.2, S.25

Begriff 21: Gruppenwerkzeug

Ein Werkzeug, das auf einem Material in der gemeinsamen Umgebung arbeitet, ist ein Gruppenwerkzeug.

Gruppenwerkzeuge vergegenständlichen wiederkehrende Kooperationssituationen im Rahmen einer bestimmten Aufgabe. Sie realisieren ein spezielles Kooperationsmodell.

Technisch löst ein Gruppenwerkzeug die Probleme der Verteilung und der Nebenläufigkeit.

Eine Handlung mit einem Gruppenwerkzeug hat für alle Benutzer – spürbar oder nicht – sofort Konsequenzen, da mit ihm Dinge in der gemeinsamen Umgebung manipuliert werden. Stellen das Gruppenwerkzeug oder dessen Kontext entsprechende Awarenessinformationen bereit, ist dies auch wahrnehmbar.

Dabei muss das Kooperationsmodell nicht unbedingt synchron sein. Wird z.B. mithilfe des Registrars ein Dokument aus einer Registratur entnommen, so steht es in diesem Augenblick allen anderen Benutzern nicht mehr zur Verfügung. Dennoch erfährt dies ein betroffener Benutzer erst, wenn er versucht, das Dokument zu entnehmen und stattdessen eine Fehlkarte erhält.
Die Kooperationsmodelle verschiedener Gruppenwerkzeuge können sehr unterschiedlich sein – je nachdem, welches Kooperationsmodell dem Zweck des Werkzeugs am besten entspricht.

Für die Bearbeitung von Dokumenten ist z.B. ein Kooperationsmodell sinnvoll, das ausschließlich die exklusive Benutzung des Dokuments vorsieht[343]: Während ein Benutzer das Dokument in der gemeinsamen Umgebung bearbeitet, ist es den anderen Benutzern dann nicht möglich, mit ihren Gruppenwerkzeugen darauf zuzugreifen.
Trotz der exklusiven Benutzung des Dokuments ist ein solches Werkzeug ein Gruppenwerkzeug, da es die Koordination in der Gruppe unterstützt.

Behälterartige Materialien, wie z.B. Räume, Registraturen, schwarze Bretter usw. gestatten, dass mehrere Benutzer parallel auf die enthaltenen Dinge zugreifen. Ein Gruppenwerkzeug zum Bearbeiten solcher Behälter muss in seinem Kooperationsmodell dafür Sorge tragen, dass die Benutzer mit unterschiedlichen Dingen arbeiten können, ohne sich "in die Quere zu kommen". Der RoomExplorer realisiert das z.B., indem von Benutzern markierte Dinge für diese exklusiv reserviert werden und dies über Awarenessinformationen publiziert wird.

Daneben sind auch Kooperationsmodelle denkbar, in denen Benutzer z.B. nach bestimmten Regeln abwechselnd auf gemeinsame Dinge zugreifen (z.B. bei Brettspielen oder beim Chat) oder gemeinsame Dinge tatsächlich gemeinsam benutzen (z.B. eine Zeichenfläche).

[343] Ein Gruppenwerkzeug, das mehreren Benutzern gleichzeitig die Bearbeitung eines Dokuments gestattet, ist nicht unbedingt zweckdienlich. Dennoch gibt es z.B. Groupware-Produkte zum synchronen Editieren von Texten. Die Chancen, sich "in die Quere zu kommen", sind allerdings sehr hoch. Man stelle sich vor, wie eine Änderung an einem Text z.B. den Seitenumbruch für ein ganzen Dokument verändert. Dies würde alle Mitbenutzer betreffen und in ihrer Arbeit beeinflussen.

Explizite Kooperationsmedien, wie z.B. das Postversandsystem oder ein Chat-Tool, dienen nicht unmittelbar der Arbeit an einem Material in der gemeinsamen Umgebung (obwohl z.B. das Postversandsystem eine Postfachwand[344] innerhalb der gemeinsamen Umgebung benutzt). Dennoch zählen auch sie zu den Gruppenwerkzeugen.

Der RoomExplorer (der, wie gesagt, selbst ein Gruppenwerkzeug ist) bildet für die übrigen Gruppenwerkzeuge einen Kontext – ähnlich wie der Desktop für die "Einzel-Werkzeuge".
Für eine erweiterte Raumkomponente ist dabei zu überlegen, wie der RoomExplorer das Starten von Gruppenwerkzeugen ermöglichen kann. Es ist z.B. vorstellbar, dass der RoomExplorer automatisch das geeignete Standard-Gruppenwerkzeug startet, wenn ein Benutzer z.B. doppelt auf ein Material in einem der dargestellten Räume klickt. Andere, "kleinere" Gruppenwerkzeuge, z.B. zum Lesen einer Notiz in einem Raum, können in Form eines Kontextmenüs präsentiert werden, das z.B. per rechtem Mausklick direkt über einem Ding-Icon des RoomExplorer geöffnet werden kann. Mit solchen Gruppenwerkzeugen, die Subwerkzeuge des RoomExplorers sind, könnte die direkte Manipulation von Dingen im RoomExplorer über das bloße Bewegen hinaus erweitert werden (z.B. um ein Ding zum Inventar hinzuzufügen o.ä.).

Zudem erscheint mir sinnvoll, dass der RoomExplorer einer erweiterten Raumkomponente allgemeine Awarenessinformationen anzeigt, z.B. welche Benutzer mit welchen Gruppenwerkzeugen auf welchen Materialien arbeiten.

All dies bietet der RoomExplorer der Basis-Raumkomponente noch nicht.

Um ein Beispiel für die technische Realisierung eines Gruppenwerkzeugs zu geben, verweise ich auf das Kapitel zur Konstruktion des RoomExplorers[345].

3.4.2 Eingebettete Plätze in Räumen: Möbel und Stapel

Die Räume der physischen Arbeitsumgebung sind i.a. mit verschiedenen Dingen möbliert. So finden sich darin Stühle, Lampen, Tische, schwarze Bretter, Aktenschränke, Regale, Papierkörbe usw.

Nicht alle Möbel sind davon für virtuelle Räume sinnvoll, wie z.B. Stühle oder Lampen.
Allen übrigen Möbeln ist gemeinsam, dass sie Dinge bereitstellen oder aufnehmen.
Bereits der herkömmliche Einzelarbeitsplatz nach WAM ist mit solchen Möbeln ausgestattet: Dort finden sich z.B. Registraturen oder ein Papierkorb, aber auch Dinge, wie z.B. ein Werkzeug-Kasten, ein Ordner mit Materialvorlagen, Postkörbe usw., die man im weiteren Sinne als "Möbel" auffassen kann[346].

---

[344] die Postfachwand ist ein behälterartiges Material, in dem vorübergehend nicht zustellbare Dinge zwischengelagert werden ([Freund00], S.83)
[345] Kap.4.2, S.129
[346] Die Vorstellung von einem Einzelarbeitsplatz als Schreibtisch – so wie durch die Desktopmetapher vorgesehen – "beißt" sich mit der Vorstellung einer Möblierung desselben.

Mithilfe von Möbeln kann Ordnung in die gemeinsame Umgebung gebracht werden. Einige Möbel, wie z.B. schwarze Bretter oder Aktenschränke, dienen darüberhinaus als implizite Kooperationsmedien. Mithilfe des Inventarkonzepts können Möbel fest an Räume gebunden werden. Die beliebig nutzbaren "Allzweck-Räume" der Basis-Raumkomponente lassen sich so für einen bestimmten Zweck ausstatten.

Fast alle Dinge der physischen Welt lassen sich auf eine Weise stapeln oder anhäufen. Neben den Möbeln dienen auch Stapel dem Aufnehmen und Bereitstellen von Dingen.

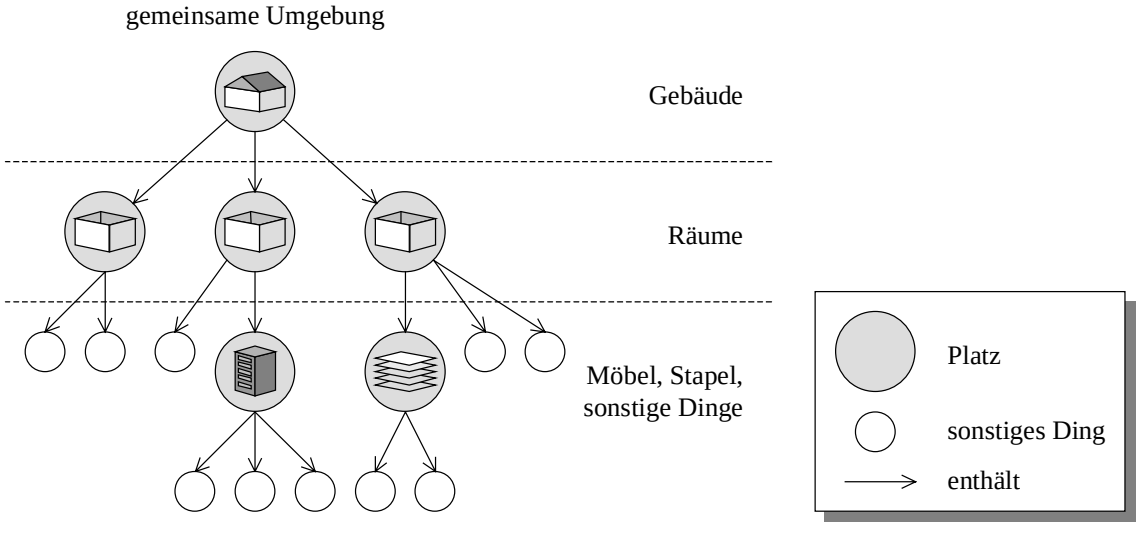
Im Umgang unterscheiden sich Stapel grundlegend von Möbeln. So können sie z.B. keine weiteren Stapel enthalten, sondern sich mit diesen lediglich zu einem neuen Stapel vereinigen. Zudem sind sie von Natur aus weniger beständig, d.h. Stapel sind keine Stapel mehr, wenn sie weniger als zwei Elemente enthalten[347]. Demnach können Stapel auch nicht inventarisiert werden.

Geeignete Möbel, wie z.B. Tische, können Stapel enthalten. Möbel sind jedoch nicht möbliert, d.h. sie können keine anderen Möbel enthalten.

3.4.2.1 Die Platz-Hierarchie der gemeinsamen Umgebung

Betrachtet man sämtliche Plätze der gemeinsamen Umgebung, ergibt sich folgendes Bild: Ein Gebäude besteht ausschließlich aus Räumen. Räume enthalten Möbel, Stapel und sonstige Dinge, können aber keine anderen Räume enthalten. Möbel können Stapel und sonstige Dinge beinhalten, aber keine anderen Möbel. Stapel wiederum enthalten ausschließlich sonstige Dinge. Gewöhnliche Behälter können zudem keine Plätze – also Gebäude, Räume, Möbel oder Stapel – enthalten.

Daraus ergibt sich eine festgelegte Hierarchie in der Baumstruktur des fachlichen Modells der gemeinsamen Umgebung:



[347] dann handelt es sich einfach um ein einzelnes Ding und der Platz, der existierte, als auf diesem Ding noch weitere lagen, ist nicht mehr vorhanden

Abbildung 23: Die Platz-Hierarchie der gemeinsamen Umgebung

Plätze sind in der Baumstruktur des fachlichen Modells der gemeinsamen Umgebung die zur Wurzel hin abschließenden Elemente.

Diese Hierarchie der Plätze dient der Orientierung im System und unterstützt die Benutzer beim Bilden, Aufrechterhalten und Befolgen von Konventionen.
Im Gegensatz dazu stehen z.B. die POLIAwaC-Arbeitsbereiche[348], die – ausgehend vom gemeinsamen Desktop – beliebig tief und in beliebiger Reihenfolge geschachtelt werden können. Die Benutzer der POLIAwaC-Arbeitsbereiche werden diese zwar unterschiedlich verwenden, jedoch wird diese qualitative Unterscheidung im System nicht explizit modelliert. Eine Analogie zwischen den Plätzen der physischen Umgebung und den virtuellen Arbeitsbereichen ist nur eingeschränkt gegeben. Die Übertragung bestimmter Konventionen, die an Plätze der physischen Arbeitswelt geknüpft sind, scheint dadurch beliebig möglich zu sein und wird durch die POLIAwaC-Arbeitsbereiche nicht motiviert. Die Benutzer können sich nicht an ihren Erfahrungen mit der bewährten physischen Umgebung orientieren[349].

Die COJAC Raumkomponente ermöglicht dagegen, den Entwurf der virtuellen Plätze an der physischen Umgebung zu orientieren. Spezielle Möbel können realisiert und die Räume damit ausgestattet werden. Auf diese Weise kann eine virtuelle Umgebung bereitgestellt werden, die den Anforderungen und Erwartungen der Benutzer entspricht. Diese können relevante Konventionen und Erfahrungen mit der physischen Umgebung in die virtuelle Umgebung übernehmen.

[348] Kap.1.3.3, S.34 – die Verzeichnisse eines Dateisystems sind ein ähnliches Beispiel
[349] Die Autoren weisen bzgl. privater und öffentlicher Bereiche darauf hin, dass Konventionen aus der Zeit vor der Einführung von POLITeam leicht übernommen werden konnten, wenn die Analogie klar war. ([Mark97], S.260) Dies gilt meiner Meinung nach nicht nur für die Unterscheidung von privaten und öffentlichen Bereichen, sondern allgemein für die Unterscheidung von Plätzen bzgl. beliebiger, fachlich relevanter Eigenschaften.

3.4.2.2 Die Handhabung von eingebetteten Plätzen im RoomExplorer

Das Benutzungsmodell des RoomExplorers der Basis-Raumkomponente berücksichtigt bisher nur die Handhabung und Präsentation des Gebäudes und der Räume in besonderer Weise. Weitere Plätze, wie z.B. Registraturen, sind lediglich wie alle anderen Dinge in den Räumen auch zu benutzen. Für die Handhabung von einigen Möbeln, wie z.B. der Registratur, mögen die existierenden Gruppenwerkzeuge, wie der Registrar, dabei genügen.

Für Möbel mit einer Arbeitsfläche, die desktop-ähnlich visualisiert werden kann, sollte das Benutzungsmodell des RoomExplorers m.E. ausgedehnt werden. Solche Möbel sollten wie die Räume direkt manipuliert werden können.
Hier ist z.B. eine Art Kontextmenü vorstellbar. Durch einen rechten Mausklick auf das Icon eines Möbels kann an der Mausposition ein Menü-Panel geöffnet werden, in dem der Inhalt der Arbeitsfläche desktop-artig visualisiert wird[350]. Durch Drag&Drop kann ein Benutzer dann Dinge aus diesem Platz entnehmen, ohne ein spezielles Werkzeug öffnen zu müssen. Ebenso kann sich dieses Menü-Panel öffnen, wenn ein Ding via Drag&Drop über das Icon des Möbelstücks gezogen wird.

Bei der direkten Manipulation von Stapeln ist u.U. eine "Stack-Semantik" für die Handhabung hilfreich (siehe Abb.24). Über Drag&Drop kann z.B. ein Ding auf den Stapel draufgelegt werden – entspricht dem "Push" der "Stack-Semantik". Ensprechend kann das zuoberst liegende Ding per Drag&Drop vom Stapel genommen und an anderer Stelle abgelegt werden – "Pop" (A). Das Icon des zuoberst liegenden Dinges kann mit einem Stapel-Icon kombiniert werden, so dass erkennbar ist, welches Ding obenauf liegt – "Top" (B).

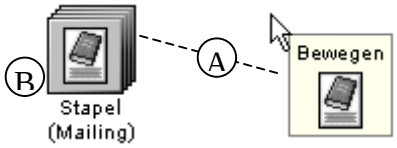


Abbildung 24: Vision zur direkten Manipulation eines virtuellen Stapels

Um einen Stapel nach bestimmten Dingen zu durchsuchen, muss auf ein spezielles Werkzeug zurückgegriffen werden. Oder auch hier kommt wie bei den Möbeln eine Art Kontextmenü zum Einsatz, das den Stapel z.B. in Form einer Liste visualisiert.

3.4.3 Abschließbare Räume

In der Basis-Raumkomponente werden keine Zugriffsrechte modelliert. Somit kann der eingeschränkte Zugriff auf einen Raum vorerst nur über die Einhaltung von Konventionen realisiert werden. Die Erweiterung der Basis-Raumkomponente um abschließbare Räume bietet dagegen einen echten Zugriffsschutz.

[350] ähnlich dem Gebäudeplan-Menü, in dem der Inhalt eines Raumes auf diese Weise angezeigt wird

Konzept	Zweck
eingeschränkter Zugriff auf die Plätze der gemeinsamen Umgebung[351]	für Handlungssicherheit in unterschiedlich starkem Maße

In [Beis00] entwickelt der Autor ein Schlüssel-Schloss-Konzept für das JWAM-Framework. Die Idee dahinter ist intuitiv verständlich:

- Sicherheitsrelevante Teile eines Systems sind mit einem Schloss versehen.
- Benutzer verfügen über Schlüssel, mit denen sie Schlösser öffnen können.

Für die erweiterte Raumkomponente bedeutet das, Räume mit Schlössern zu versehen, sofern sie nicht vollkommen öffentlich sind. Die Abteilungsleiterin aus dem Buchverlagsbeispiel kann ihren persönlichen Arbeitsraum dann z.B. abschließen. Andere Benutzer können abgeschlossene Räume weder fokussieren, noch in sie hineinsehen.

Abschließbare Räume müssen im Benutzungsmodell des RoomExplorers berücksichtigt werden (siehe Abb.25). Der Bereich eines RoomTops, in dem normalerweise die Fußbodenfläche mit den darauf befindlichen Dingen visualisiert wird, bleibt z.B. frei, wenn der entsprechende Raum abgeschlossen ist, und der Benutzer keinen Schlüssel zu dem Raum besitzt (A). Desgleichen wird keine Awarenessinformation darüber angezeigt, wer sich in dem abgeschlossenen Raum befindet[352].

Abbildung 25: Vision zu abschließbaren Räumen

Benutzer, die einen Schlüssel besitzen, können den abgeschlossenen Raum handhaben, als wäre er nicht abgeschlossen. Sie sehen z.B. an einem Icon in der Titelzeile des RoomTops, ob der Raum auf- oder abgeschlossen ist.

[351] Kap.2.2.3, Tabelle 5, S.67

[352] Evtl. könnte eine Art "Klingelknopf" angezeigt werden, der gedrückt werden kann, um den in dem Raum befindlichen Personen zu signalisieren, dass man den Raum betreten möchte.

In der physischen Welt wird ein Raum implizit abgeschlossen, indem man alle seine Türen abschließt. Die virtuellen Räume der erweiterten Raumkomponente werden dagegen explizit abgeschlossen[353].

3.4.4 Gruppieren von Räumen: Etagen und Trakte

Die starre Topologie des Gebäudes kann in der Basis-Raumkomponente nicht explizit strukturiert werden, d.h. Räume können nicht explizit gruppiert werden. Auch hier besteht nur die Möglichkeit, durch bestimmte Eigenschaften der Räume diese Struktur implizit zu modellieren (z.B. indem die Räume der einen Abteilung einen roten, die der anderen einen blauen "Teppichboden" haben). Dabei sind aus der physischen Welt z.B. Etagen und Trakte usw. bekannt, die die Struktur eines Gebäudes prägen.

Trakte sind z.B. sinnvoll, um eine zusammenhängende Gruppe von Räumen im Ganzen abschließen zu können. In Hinblick auf die bereits angesprochene survey knowledge bei der räumlichen Orientierung[354] ist so explizit klar, welche Bereiche verschlossen sind. Durch abschließbare Türen werden lediglich Wege abgeschnitten – dadurch können ganze Bereiche unzugänglich sein, aber nur implizit, es können aber auch Bereiche weiterhin über andere Wege zugänglich sein, so dass eine abgeschlossene Tür keinen Effekt hat.

Visualisiert werden können Trakte z.B. über einheitlich eingefärbte Räume, so wie es in der Basis-Raumkomponente schon informal realisiert werden konnte (siehe Abb.26[355]).

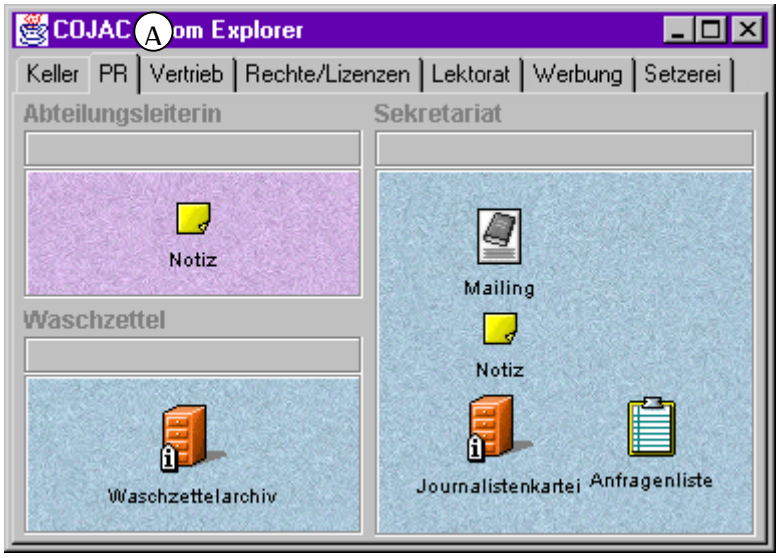


Abbildung 26: Vision zur Visualisierung von Etagen

Etagen bieten die Möglichkeit, große Gebäude zu realisieren – in der physischen wie in der virtuellen Welt. Einzelne Etagen der erweiterten Raumkomponente sind aufgebaut, wie ein einzelnes Gebäude der Basis-Raumkomponente. Die einzelnen Etagen müssen in ihrem

[353] Um hierzu eine Vision des Benutzungsmodells geben zu können, kenne ich die neue JWAM-Schlüsselkomponente noch nicht gut genug.
[354] Kap.3.2.3.6, S.87
[355] in der Abbildung bilden Sekretariat und Waschzettelraum einen Trakt

Grundriss nicht unbedingt übereinanderpassen. Sie können also unabhängig voneinander gestaltet werden.

Denkbar sind eine "Kelleretage" für große, zentrale Automaten, wie z.B. die Postdienststelle des Postversandsystems[356], und Etagen für einzelne Abteilungen.
Auch Etagen sind abschließbar. So kann die Kelleretage z.B. nur vom Systemadministrator betreten werden – dem "Hausmeister" des virtuellen Gebäudes – um Einstellungen an den Automaten vorzunehmen.

Die einzelnen Etagen eines Gebäudes befinden sich auf demselben Server[357].

Visualisiert werden Etagen passenderweise übereinander in "Schichten" (siehe Abb.26). Dies passt zu der Grundrissansicht des gesamten Gebäudes. Über ein Reiter-Widget kann eine einzelne Etage in den Vordergrund geholt werden (A). Es ist immer nur eine Etage sichtbar – das entspricht der Erfahrung aus physischen Gebäuden.

3.4.5 Explizite synchrone Kommunikation in Räumen: Chat- / Audio-Verbindung

Konkurrenzsituationen werden zwar mithilfe des RoomExplorers erkannt, jedoch können sich die betroffenen Kooperationspartner nicht mithilfe des RoomExplorers koordinieren, da ihnen die Möglichkeit zur expliziten synchronen Kommunikation fehlt.
Arbeiten die betroffenen Benutzer in ein und demselben physischen Raum, stellt das kein Problem dar, weil mündlich kommuniziert werden kann. Ist dies nicht der Fall, kann ein Chat-Tool oder eine Audio-Verbindung diesen Mangel beheben.

Die explizite synchrone Kommunikationunterstützung ist an den Raum gebunden, d.h. die Chat- bzw. Audio-Verbindung wird aufgebaut, sobald ein Benutzer einen Raum fokussiert. In Analogie zur physischen Welt können auf diese Weise alle Benutzer, die denselben Raum fokussieren, sofort miteinander explizit kommunizieren.

Diese Lösung wird z.B. auch von Orbit und TeamWave in vergleichbarer Weise realisiert. Vor allem die TeamWave-Lösung dient hier als Vorbild[358]. Orbit bietet eine Audio-Video-Verbindung zwischen Benutzern, die dieselbe Sicht auf gemeinsame Orte eingestellt haben[359].

---

[356] [Freund00], S.80
[357] es ist also wie in der Basis-Raumkomponente das Gebäude, über das die technische Verteilung fachlich transparent gemacht wird
[358] Kap.1.3.2, Abb.4 (C), S.33
[359] Kap.1.3.2, Abb.3 (C), S.30

Abbildung 27: Vision zum Chat-Tool in der Raumansicht

Bewegt ein Benutzer in der Gebäudeansicht des RoomExplorers Dinge von einem Raum in den anderen, ist es vermutlich störend, wenn ständig die Chat- bzw. Audio-Verbindung gewechselt wird. Es liegt daher nahe, die Verbindung nur aufzubauen, wenn der Benutzer seinen RoomExplorer in die Raumansicht umschaltet.

Ein Chat-Tool könnte z.B. im RoomExplorer unterhalb des in Raumansicht geschalteten RoomTops als Subwerkzeug eingebettet werden (siehe Abb.27). Durch besondere Smileys kann dargestellt werden, ob ein Benutzer eine Chat-Verbindung aufgebaut hat und ob er z.B. gerade chattet (A)[360].

Durch die Verknüpfung der Chat- bzw. Audio-Verbindung mit der Raumansicht kann ein Benutzer die Verbindung explizit unterbrechen, indem er in die Gebäudeansicht schaltet.

3.4.6 Gebäudeplan-Menü

Ein ergonomisches Problem, das sich bei vielen Desktop-Systemen zeigt, ist, dass die GUI-Frames der geöffneten Werkzeuge den Desktop verdecken. Um auf die Funktionen des Desktops zugreifen zu können, müssen diese Frames erst beiseite geschoben oder minimiert werden. Gleiches gilt für den JWAM-Desktop und den RoomExplorer, die von den geöffneten Werkzeug-Frames verdeckt werden.

Ein Gebäudeplan-Menü kann da Abhilfe schaffen (siehe Abb.28). Dieses stellt in einem Menü-Panel zunächst den Gebäudeplan dar (A). Wählt man darin einen Raum aus, öffnet sich in einem weiteren Menü-Panel der RoomTop dieses Raumes (B)[361].

---

[360] Die Benutzer mit Verbindung haben hier einen roten Smiley, die anderen den "normalen" gelben. Der Smiley des Benutzers, der die letzte Chat-Nachricht geschrieben hat, hat einen geöffneten Mund.

[361] Das realisierte RoomTop-Werkzeug (siehe Kap.4.2.4, S.43) eignet sich, um in ein solches Menü als Subwerkzeug eingebettet zu werden.

Ein solches Menü kann als Subwerkzeug in Gruppenwerkzeuge integriert werden. Ein entsprechendes Menü für die private Umgebung ist für "Einzelwerkzeuge" sicher ähnlich sinnvoll.

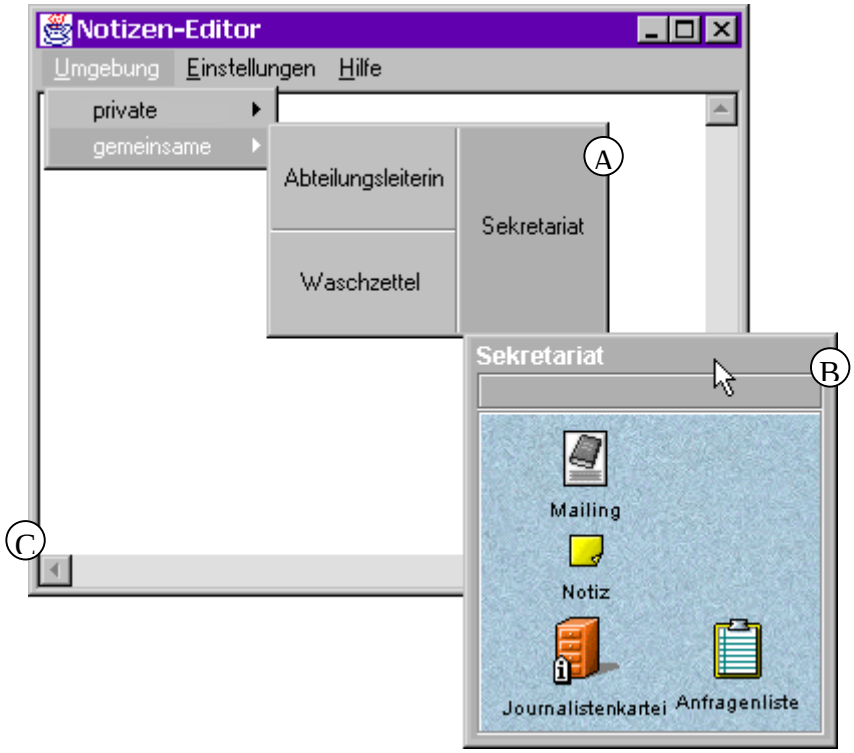


Abbildung 28: Vision zum Gebäudeplan-Menü

Dinge können im aufgeklappten RoomTop doppelt angeklickt und so zur Bearbeitung ausgewählt werden – sofern sich das Werkzeug zur Bearbeitung des gewählten Dinges eignet[362].
Das Gebäudeplan-Menü ist sozusagen das Pendant zum "Datei öffnen"-Dialog z.B. der Win95-Applikationen[363].

Ein Problem ist möglicherweise, dass durch mehrere Gebäudeplan-Menüs in verschiedenen Werkzeugen das Gebäude an mehreren Stellen des Systems visualisiert wird. Das Verständnis der Einheit von Zeit und Raum kann dadurch beeinträchtigt werden. Ein einzelner RoomExplorer vergegenständlicht das Gebäude. Mehrere Gebäudeplan-Menüs sind dagegen virtuelle Sichten auf das Gebäude, verbunden mit einer gewissen Beliebigkeit und Mehrdeutigkeit.

[362] Falls nicht, kann z.B. das entsprechende Standardwerkzeug für das gewählte Ding gestartet werden. Insofern ist es sinnvoll, in allen Werkzeugen Menüs sowohl für die private als auch die gemeinsame Umgebung anzubieten.
[363] Dinge verbleiben i.d.R. an dem Ort, an dem sie bearbeitet werden, somit ist es nicht notwendig, sie nach der Bearbeitung über das Gebäudeplan-Menü wieder zurückzulegen. Es ist Sache des Werkzeugs, auf welche Weise die Bearbeitung des Materials beendet wird.

3.4.7 Web-Interface

Ein Web-Interface kann den Zugriff auf gemeinsame Räume über einen Webserver ermöglichen. Auf diese Weise ist keine besondere Software erforderlich, ein beliebiger Internet-Browser genügt. Dinge können von Raum zu Raum verschoben und zwischen lokalem Rechner und gemeinsamem Gebäude via Download/Upload ausgetauscht werden.

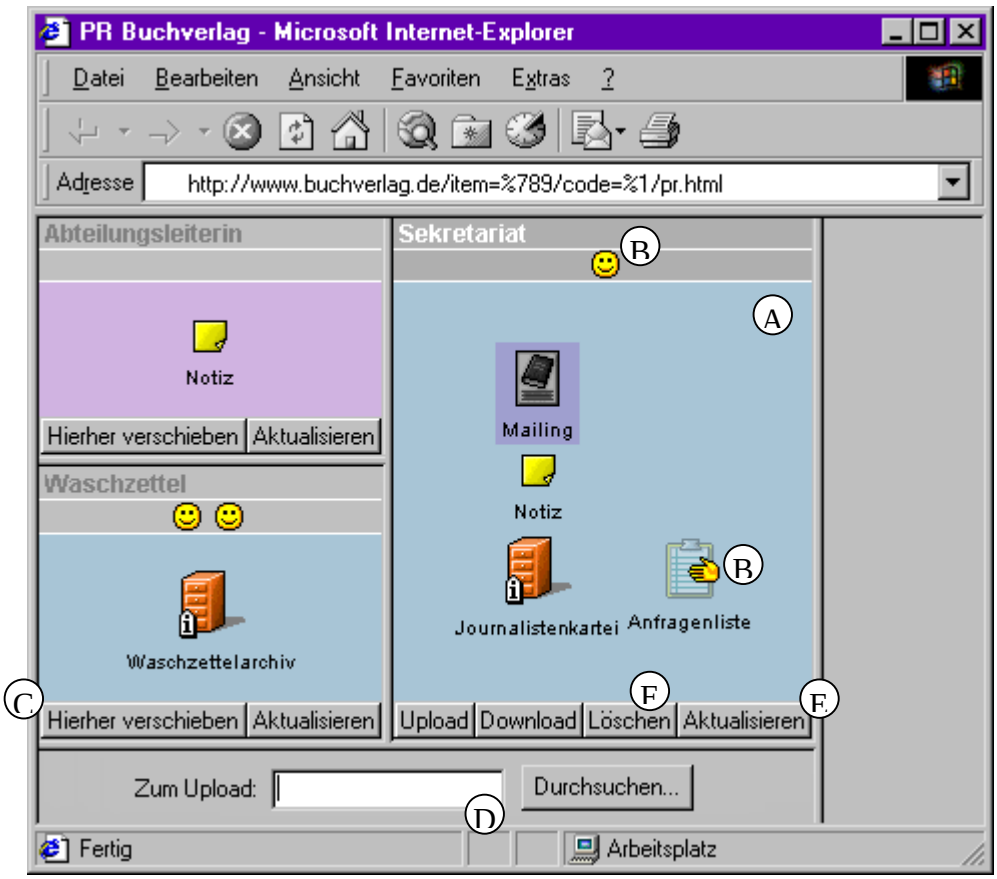


Abbildung 29: Vision zum Web-Interface[364]

Der Webserver erzeugt dynamisch Webseiten für die einzelnen Räume, die deren Zustand widerspiegeln.
Mit den Mitteln von HTML können diese den RoomTops des RoomExplorers ähnlich visualisiert werden (siehe Abb.29). So enthalten die Räume z.B. in einer (unsichtbaren) Tabelle positionierte Icons als Repäsentationen der Dinge, die auf ihrer Fußbodenfläche angeordnet sind (A). Auch die Awarenessinformationen, wie die anderen Benutzer und deren markierte Dinge, werden in gleicher Weise dargestellt (B).

Das Markieren von Dingen geschieht wie im RoomExplorer über das Anklicken des entsprechenden Icons. Ebenso können Räume Fokussiert werden, indem z.B. auf deren Titelzeile geklickt wird.

[364] Bei dem hier gezeigten Internet-Browser handelt es sich um den Microsoft® Internet-Explorer™, © 1995-1999 Microsoft Corp.

Im Web-Interface steht im Ggs. zum RoomExplorer kein Drag&Drop für das Bewegen von Dingen zur Verfügung. Daher werden die entsprechenden Aktionen per Knopfdruck ausgeführt.

Um ein Ding von einem Raum in den anderen zu Verschieben, klickt der Benutzer z.B. zunächst das entsprechende Icon im Web-Interface an. Das Ding wird auf dem Web-Server für den Benutzer reserviert und die Darstellung des Raumes aktualisiert. Das Ding erscheint daraufhin markiert. Nun kann der Benutzer das Ding in einen anderen Raum verschieben, indem er auf den entsprechenden Knopf in der Fußzeile des Raumes klickt (C). Innerhalb eines Raumes können Dinge ohne Drag&Drop nicht verschoben werden. Sie werden automatisch an freie Positionen gelegt.
Ähnlich funktioniert der Download von Dingen. Zunächst wird ein Ding markiert und dann per Knopfdruck in der Fußzeile des fokussierten Raumes auf den lokalen Rechner geladen. Das geschieht wie üblich über den Download-Dialog des Internet-Browsers.
Zum Upload trägt der Benutzer einen Dateinamen in ein spezielles Textfeld ein (D). Über einen Knopf in der Fußzeile des Raumes, in den die Datei kopiert werden soll, startet er den Upload-Vorgang.

Werden Frames unterstützt, so können Räume in einzelnen Frames angezeigt werden, die sich unabhängig voneinander aktualisieren lassen. Werden keine Frames unterstützt, muss das ganze Gebäude durch eine einzelne HTML-Seite visualisiert werden.

Über einen Polling-Mechanismus wird die Darstellung des Gebäudes regelmäßig aktualisiert[365]. Zudem gehört es zur normalen Browserfunktionalität, die Darstellung auf Knopfdruck aktualisieren zu lassen. Werden Frames verwendet, wird ein spezieller Knopf zum Aktualisieren eines einzelnen Raumes in dessen Fußzeile angeboten (E).

Einige wichtige Fragen zum Entwurf des Web-Interfaces bleiben hier offen:

So ist unklar, auf welche Weise sich die Einheit von Raum und Zeit bei der Benutzung des Web-Interface aufrechterhalten lässt. Lädt der Benutzer z.B. über das Web-Interface eine Datei in die gemeinsame Umgebung hoch, so wird diese nicht automatisch gelöscht, wenn der Upload-Vorgang erfolgreich war. Ähnliches gilt für den Download-Vorgang.
Das Up- und Downloaden von Dingen erzeugt stets fachliche Kopien der Dinge. Überflüssige fachliche Kopien müssen vom Benutzer explizit gelöscht werden, z.B. indem er nach einem erfolgreichen Download-Vorgang das noch immer markierte Ding per Knopfdruck aus dem fokussierten Raum löscht (F).

Offen bleibt zudem, ob sämtliche Dinge über das Web-Interface austauschbar sein sollten, oder z.B. nur Dokumente, die sich auch außerhalb eines WAM-Systems verwenden lassen. Die austauschbaren Dokumente müssten besonders gekennzeichnet werden.

[365] Ähnlich funktioniert z.B. oft die Aktualisierung von WebCam-Bildern, die über das Internet einen Blick auf die Kaffemaschine im Nachbarzimmer o.ä. gewähren.

Eine sinnvolle Erweiterung des Web-Interfaces wären Gruppenwerkzeuge, die ebenfalls als Web-Interface gestartet werden können. Mit diesen könnten z.B. Dokumente auf dem Web-Server angezeigt werden.

4. Konstruktion der COJAC Basis-Raumkomponente

In diesem Kapitel möchte ich schließlich die Konstruktion und Implementation der Basis-Raumkomponente auf der Grundlage des JWAM-Frameworks beschreiben.

Die verwendete Version ist JWAM 1.5alpha (ca. April 2000). Begonnen hatte ich mit der Version 1.4.1 (ca. Februar 2000).
Seit September 2000 gibt es die Version 1.5.0, die sich gegenüber der 1.5alpha in einigen wichtigen Teilen geändert hat (z.B. im User-Management, in der Vererbungsschnittstelle der abstrakten Fachwertklasse, der Package-Struktur[366] usw.). Diese Änderungen habe ich in meiner Implementation noch nicht berücksichtigt.

Auf der anderen Seite musste ich einige wenige Änderungen an JWAM vornehmen. So haben z.B. die Drag&Drop-Funktionalität und die Desktop-Komponente das Bewegen von Dingen vom RoomExplorer auf den Desktop nicht geeignet unterstützt[367].

Gegenüber der Version 1.4 wurde in der Version 1.5alpha die Werkzeugkonstruktion stark verändert. Subwerkzeuge haben nun denselben Aufbau wie Kontextwerkzeuge. Die Werkzeuge RoomExplorer und RoomTop entsprechen bereits dieser neuen Konstruktionsweise[368].
Im Anschluss an die Fertigstellung der Arbeit ist geplant, die Raumkomponente in die aktuelle Version des JWAM-Frameworks einzubinden.

Die Konstruktion der Basis-Raumkomponente basiert im wesentlichen auf Schnittstellen. Die Schnittstellen definieren die Hot-Spots[369] der Basis-Raumkomponente.
Die Semantik der einzelnen Operation der Schnittstelle spezifiziere ich über Vor- und Nachbedingungen. Dies erfolgt in JWAM über Kommentare, die zu jeder Signatur der Schnittstelle die jeweiligen booleschen Ausdrücke der Vor- und Nachbedingungen beschreiben.

Zu jeder dieser Schnittstellen biete ich in einer korrespondierenden Klasse eine black- bzw. white-box-artig verwendbare Implementation. Als Klassenbezeichner dient der Schnittstellenbezeichner, an den der Zusatz "Impl" angehängt wird. Diese Benennungskonvention entspricht dem JWAM-Code-Styleguide[370].

---

[366] So befindet sich z.B. das Package desktop nun in jwamx statt jwamalpha – siehe Kap.1.4.3, Abb.7, S.53 Das User-Management – ursprünglich im Package de.jwam.technology.user angesiedelt – wurde in [Beis00] grundlegend überarbeitet und findet sich nun unter de.jwam.handling.accesscontrol wieder.
[367] In der bisherigen Lösung besaß ein sog. DragBoard vorübergehend eine Referenz auf das via Drag&Drop bewegte Ding. Das ist bei einem lokal nicht vorhandenen Ding nicht möglich. Daher wird es nun erst am Ende des Drag&Drop-Vorgangs ermittelt. Dem DragBoard ist während des Vorgangs lediglich die Dingbeschreibung bekannt, die es für die Anzeige des semantischen Feedbacks (Kap.3.3.6, S.102) benötigt.
[368] dazu mehr in Kap.4.2.4, S.142
[369] Kap.1.4.1, S.45
[370] [JWAM o.J.]

Durch die Trennung von Schnittstelle und Implementation wird die lose Kopplung gefördert. Die Klassen zu den Schnittstellen werden in anderen Klassen nie direkt, sondern nur über die implementierte Schnittstelle verwendet.
Die einzige Ausnahme von der Trennung in Schnittstelle und Implementation bilden einige Fachwert- und Hilfsklassen, deren Konstruktionsweise bereits durch JWAM vorgegeben ist.

Die bei den Operationen der Schnittstelle spezifizierten Vor- und Nachbedingungen werden – wie in JWAM üblich – jeweils zu Beginn und am Ende einer Operation geprüft. Dazu verwende ich die Klassen des in der JWAM-Spracherweiterungsschicht implementierten Vertragsmodells[371].

Darüberhinaus existiert zu jeder Klasse eine Testklasse, in der zu den Operationen der Klasse verschiedene Testfälle implementiert werden.
Testklassen werden in JWAM mit den eigentlichen Klassen kontinuierlich weiterentwickelt und laufend um neue Tests ergänzt. Über den ebenfalls in der Spracherweiterungsschicht angesiedelten Test-Mechanismus[372] können die in den Testklassen definierten Tests leicht für einzelne Packages oder das ganze JWAM-Framework ausgeführt werden.
Die Klassen der Basis-Raumkomponente wurden während der Entwicklung kontinuierlich getestet. Die implementierten Tests haben sie fehlerfrei durchlaufen.

Im Groben besteht die Basis-Raumkomponente aus zwei Teilen: dem technischen Modell der gemeinsamen Umgebung und dem RoomExplorer-Gruppenwerkzeug, mit dem auf die Räumlichkeiten der gemeinsamen Umgebung zugegriffen werden kann.
Beide Teile sind in der Handhabungs- und Präsentationsschicht angesiedelt. Das technische Modell der gemeinsamen Umgebung und der Räumlichkeiten ist im Package de.jwam.handling.environment, das RoomExplorer-Gruppenwerkzeug im Package de.jwamx.handling.roomexplorer angesiedelt[373].

Dieser Aufteilung entsprechen die zwei Teile dieses Kapitels: im ersten Teil beschreibe ich das technische Modell der gemeinsamen Umgebung, im zweiten das Gruppenwerkzeug RoomExplorer.

In jedem dieser Teile beschreibe ich in einzelnen Unterkapiteln die definierten Schnittstellen mit ihren Implementierungen sowie den konstruierten Fachwerten und Hilfsklassen[374]. Die Implementation ist recht umfangreich und es würde den Rahmen sprengen, hier jede Konstruktionsentscheidung und jedes Feature anzusprechen. Ich hoffe, mit der folgenden

[371] Kap.1.4.3, S.49 und Kap.1.4.2, S.47
[372] Kap.1.4.3, S.49
[373] D.h. in diesen Packages werden sie voraussichtlich in das JWAM-Framework integriert werden. Derzeit befinden sich die Packages environment und roomexplorer in einem von mir angelegten zusätzlichen Package namens de.jwamj.handling. Ich war davon ausgegangen, dass sich die JWAM-Package-Struktur bis zum Ende meiner Arbeit verändern würde, so dass meine Packagezuordnung ohnehin keinen Bestand gehabt hätte. Durch das jwamj-Package hatte ich während der Entwicklung alle bearbeiteten Klassen und Schnittstellen an einem Platz.
[374] z.B. zur Ausnahmebehandlung oder zum Nachrichtenversand

Beschreibung einen plausiblen, konsistenten Einblick in die Implementation geben zu können. Zudem habe ich mich bemüht, dass dieses Kapitel auch ohne JWAM-Vorkenntnisse lesbar ist.

An den Beginn eines jeden Unterkapitels stelle ich jeweils eine kurze Übersicht über die realisierten Schnittstellen und Klassen und welche Schnittstellen von ihnen erweitert bzw. implementiert werden.
Bei den realisierten Klassen gebe ich an, ob eine black-box oder white-box-artige Verwendung derselben von mir intendiert ist.
Auf Implementationsvererbung gehe ich nicht weiter ein[375], da es mir hier vor allem um eine entwurfsbezogene Beschreibung der Einbettung der Raumkomponente in JWAM geht.

4.1 Das technische Modell der gemeinsamen Umgebung

Das fachliche Modell hängt im WAM-Ansatz – wie bereits geschildert – eng mit dem technischen Modell zusammen[376]. Im Objekt-Metamodell bestehen bestimmte, festgelegte Zusammenhänge zwischen beiden Modellen. So werden Begriffe des fachlichen Modells im technischen Modell durch eine Klasse abgebildet, Gegenstände durch Objekte, Umgangsformen durch Operationen.

In diesem Sinne existiert zum Begriff "Raum" eine Klasse "Room" und zum Begriff "Gebäude" eine Klasse "Building". Verwende ich im folgenden den englischen Bezeichner, so meine ich die Klasse bzw. Schnittstelle des technischen Modells. Verwende ich den deutschen Begriff, so ist das entsprechende anwendungsfachliche Konzept gemeint.

Die Struktur des technischen Modells ist im Wesentlichen durch die Begriffshierarchie des fachlichen Modells beeinflusst. Einzelarbeitsplätze, Räume und Gebäude sind im fachlichen Modell Plätze. Erwartungsgemäß sollten die entsprechenden Klassen eine gemeinsame abstrakte Oberklasse bzw. Schnittstelle namens *Place* besitzen.
Zwar wird die Platz-Metapher durch Einzelarbeitsplätze, Räume und Gebäude konkretisiert, doch weisen diese Konzepte keine allgemeinen Merkmale im Umgang auf, die in einer gemeinsamen Schnittstelle sinnvoll "herausgeneralisiert" werden könnten, bzw. nicht bereits durch in anderen Schnittstelle modelliert wurden (z.B. der Container- oder Thing-Schnittstelle). Aus diesem Grund erschien mir eine Place-Schnittstelle zum Zeitpunkt der Konstruktion nicht notwendig[377].

---

[375] Ausnahme sind die Fachwerte, in deren Entwurfsmuster Schnittstellen- und Implementationsvererbung nicht getrennt werden, und die Hilfsklassen, für die Ähnliches gilt.
[376] Kap.1.1.2, S.16
[377] Inzwischen sehe ich das jedoch anders. An einer Place-Schnittstelle könnten z.B. die enthaltenen Plätze ermittelbar sein. Auf diese Weise könnte ausgehend vom "Wurzel-Platz" (dem Gebäude) die baumartige Platz-Struktur einer Umgebung ermittelt werden. Stammplätze von Dingen könnten über die Place-Schnittstelle referenziert werden, um z.B. Dinge beim Schließen eines Werkzeugs automatisch an ihren Platz zurückzulegen. Der Name für diese Schnittstelle wäre mit Space im übrigen besser gewählt, da die spezifizierten Eigenschaften eher zu den Merkmalen eines Ortes gehören. Zudem sind Plätze immer etwas Konkretes und durch eine (abstrakte) Schnittstelle nicht angemessen repräsentiert (Kap.1.3.6, S.42).

4.1.1 Room (Raum)

Schnittstelle: Room
Erweiterte Schnittstellen:
de.jwam.handling.containerconstruction.Container
de.jwam.handling.thing.Thing

Implementierende Klasse: RoomImpl
Verwendung der Klasse: Black-Box

Fachwert-Klassen:
dvRoomDescription (erweitert dvThingDescription)
dvTableOfPositions (erweitert dvTableOfContents)

Das fachliche Konzept des Raumes[378] modelliere ich in der Schnittstelle Room. Sie erweitert die JWAM-Schnittstelle Container, die ihrerseits Thing erweitert.
Zur Schnittstelle Room existiert mit der Klasse RoomImpl eine konkrete Standard-Implementierung.
Als Fachwerte kommen dvRoomDescription und dvTableOfPositions hinzu.

Ein Raum ist in erster Linie ein Behälter für Dinge. Daher erweitert die Schnittstelle Room die Container-Schnittstelle[379]:

```
    public boolean isAddable (Thing thing);
    public boolean isRemovable (dvIdentificator id);

    public void add (Thing thing);
    public void remove (dvIdentificator id);

    public boolean has (dvIdentificator id);
    public Thing thing (dvIdentificator id);

    public dvTableOfContents tableOfContents ();
```

Schnittstelle 1: Room – Ausschnitt: Container-Schnittstelle

In JWAM besitzen Behälter grundsätzlich ein Inhaltsverzeichnis. Es enthält für jedes Ding in dem Behälter eine Beschreibung, die dvThingDescription. Mit der Funktion tableOfContents kann das gesamte Inhaltsverzeichnis eines Raumes ermittelt werden, mit has, ob ein einzelnes, bestimmtes Ding enthalten ist. Über die Funktion thing kann auf ein einzelnes Ding im Raum zugegriffen werden. Bestimmt wird ein Ding jeweils über seinen Identifikator, den in JWAM jedes Objekt, welches die Thing-Schnittstelle implementiert, besitzt[380].

[378] Kap.3.2.3.2, S.82
[379] hier ausschnittsweise wiedergegeben – siehe Package de.jwam.handling.containerconstruction
[380] Dies wird an vielen Stellen des JWAM-Frameworks so gehandhabt. Der Identifikator kann z.B. aus dem Inhaltsverzeichnis eines Behälters oder aus der Dingbeschreibung eines Dinges ermittelt werden. Über den Identifikator-Fachwert kann ein Ding z.B. aus einem Behälter entfernt werden, ohne dass das Ding-Objekt selbst

Interessant sind vor allem die ersten beiden Test-Operationen isAddable und isRemovable. Beide Tests sind jeweils Vorbedingungen für die (fast) gleichnamigen Operationen add bzw. remove zum Einfügen bzw. Entfernen von Dingen.
Über die Implementierung von isAddable schließe ich in RoomImpl aus, dass Räume in einen Raum eingefügt werden können. Technisch wäre dies ohne weiteres möglich, da Room die Thing-Schnittstelle erweitert. Es gilt aber stets: `room1.isAddable(room2) == false`[381].
Mit der Implementierung von isRemovable verhindere ich, dass inventarisierte Dinge aus einem Raum entfernt werden können.

Dazu möchte ich zunächst die Operationen der Room-Schnittstelle betrachten, die sich auf das Inventar eines Raumes beziehen:

```
public void addToInventory (dvIdentificator id);
public void removeFromInventory (dvIdentificator id);

public boolean isPartOfInventory (dvIdentificator id);

public dvTableOfContents inventoryList ();
```

Schnittstelle 2: Room – Ausschnitt: Inventar-Schnittstelle

Es fällt die Analogie zur Container-Schnittstelle auf. An anderer Stelle hatte ich bereits darauf hingewiesen, dass das Inventar im abstrakten Sinne als Behälter aufgefasst werden kann[382]. Es fehlen hier jedoch die Tests zum Hinzufügen und Entfernen, da prinzipiell jedes in einem Raum enthaltene Ding inventarisiert werden kann und sämtliche inventarisierten Dinge wieder aus dem Inventar entfernt werden können.

Die Funktion inventoryList liefert das gesamte "Inhaltsverzeichnis" des Inventars.
Für die Realisierung der Inventarliste verwende ich den bereits vorhandenen JWAM-Fachwert dvTableOfContents, das Standard-Inhaltsverzeichnis eines JWAM-Containers. Dieser Fachwert muss nicht erweitert werden, um als Inventarliste zu dienen.
Die Modellierung als Fachwert ist sinnvoll, da die Inventarliste das Inventar lediglich beschreibt und nicht selbst verkörpert.

Für ein einzelnes im Raum enthaltenes Ding kann über die Funktion isPartOfInventory ermittelt werden, ob es zu dessen Inventar gehört. Ist dies der Fall, so kann das betreffende

---

bekannt ist. Bei entfernten Methodenaufrufen (z.B. über RMI) erweist es sich als vorteilhaft, dass er leichtgewichtiger ist, als das Ding selbst. M.E. liegt ein großer Nachteil der in der JWAM-Version 1.5alpha realisierten Identifikatoren darin, dass der Typ des identifizierten Dinges zur Entwurfszeit nicht festgelegt wird. Sinnvoll wären speziellere Identifikatoren, wie z.B. dvRoomIdentificator o.ä, die Subtypen von dvIdentificator sind.

[381] `room1` und `room2` sind dabei beliebige Objekte vom Typ `Room`

[382] Kap.3.2.3.5, S.86

Ding nicht aus dem Raum entfernt werden, d.h.: aus `room.isPartOfInventory(thing) == true` folgt `room.isRemovable(thing) == false`.

Die in einem Raum enthaltenen Dinge können auf seiner Fußbodenfläche angeordnet werden. Die Positionen der Dinge sind X-Y-Koordinaten in einem gedachten Raster, welches über der Fußbodenfläche liegt. In der folgenden Abbildung hätte die Anfragenliste z.B. die Position (1, 2) – der Koordinatenursprung (0, 0) ist oben links. Auf diese Weise wird von der tatsächlichen Pixel-Position bei der Darstellung abstrahiert. Die Ausdehnung der Icons muss also erst bei der Darstellung berücksichtigt werden[383].

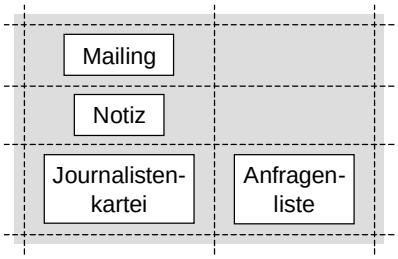


Abbildung 30: Gedachtes Raster auf der Fußbodenfläche eines Raumes

Die Ausdehnung der Fußbodenfläche entspricht der Anzahl der Zeilen und Spalten des gedachten Rasters. Die Fußbodenfläche eines Raumes ist demnach rechteckig. Je nachdem, wie die Dinge angeordnet sind, und abhängig von der Anzahl der Dinge kann die konkrete Ausdehnung des Rasters und damit der Fußbodenfläche variieren. Sondiert werden kann sie mit den Operationen width und height.

```
public int width ();
public int height ();

public Point position (dvIdentificator id);

public boolean isOccupied (Point point);
public Thing thingAt (Point point);

public dvTableOfPositions tableOfPositions ();
```

Schnittstelle 3: Room – Ausschnitt: Sondieren der Fußbodenfläche

Die Position eines Dinges auf der Fußbodenfläche wird mit der gleichnamigen Funktion ermittelt. Umgekehrt kann mit thingAt auf ein Ding, welches sich an einer bestimmten belegten Position befindet, zugegriffen werden. Die Funktion isOccupied liefert dabei zurück, ob eine Position belegt ist.

Analog zum Inhaltsverzeichnis besitzt der Raum ein Positionenverzeichnis, in dem zu jeder im Inhaltsverzeichnis aufgeführten Dingbeschreibung die Position des jeweiligen Dinges

[383] hierzu verwende ich das GridBagLayout – eine Swing-Komponente, mit deren Hilfe z.B. Icons auf einem Panel genau in der beschriebenen Weise angeordnet werden können

angegeben wird. Der Fachwert dvTableOfPositions erweitert den Fachwert dvTableOfContents um die entsprechenden Methoden, um die Positionen zu sondieren.

Eine freie Position kann mit der Methode freePosition am Raum ermittelt werden.

```
public Point freePosition ();

public void put (Thing thing, Point point);
public void move (dvIdentificator id, Point point);
```

Schnittstelle 4: Room – Ausschnitt: Positionieren von Dingen auf der Fußbodenfläche

Verwendet man die Operation add, um ein Ding in einen Raum einzufügen, so wird es an einer solchen freien Position abgelegt. Soll ein Ding an einer bestimmten Position abgelegt werden, ist die Operation put dazu geeignet. Dort kann die gewünschte freie Position als Parameter angegeben werden. Wie bei der add-Operation der Container-Schnittstelle gilt auch für die put-Operation als Vorbedingung, dass isAddable für das hinzuzufügende Ding erfüllt sein muss.
Die Operation move dient dazu, ein enthaltenes Ding an eine freie Position innerhalb desselben Raumes zu verschieben.

Abgesehen von der Fußbodenfläche und den darauf angeordneten Dingen, besitzt der Raum noch weitere Merkmale, welche in einer Raumbeschreibung zusammengefasst werden.

```
public dvThingDescription thingDescription ();
public dvRoomDescription roomDescription ();
```

Schnittstelle 5: Room – Ausschnitt: Beschreibungen eines Raumes

Die Methode thingDescription ist Teil der Thing-Schnittstelle, die von der Room-Schnittstelle erweitert und von RoomImpl implementiert wird. Der Fachwert dvThingDescription beschreibt den Raum als nicht weiter spezifiziertes Ding – er enthält den Namen, den Identifikator und ein Icon. Die dvRoomDescription erweitert den Fachwert dvThingDescription und enthält zudem noch die Farbe der Fußbodenfläche und eine Raumnummer.

Ein Icon wird für Räume nicht benötigt, da diese nicht auf dem Desktop oder in anderen Räumen abgelegt werden können. Sie werden im RoomExplorer auf spezielle Weise visualisiert. Das Icon des Fachwerts dvThingDescription dient im RoomExplorer als "Teppichboden"-Fliese, d.h. die Fußbodenfläche wird kachelartig mit dem Icon ausgefüllt[384]. Das Icon ist somit als "Teppichboden" das visuelle Erkennungsmerkmal eines Raumes.

[384] siehe Entwurf des Benutzungsmodells, Kap.3.3.1, Abb.15, S.90

Abgesehen von den in der Raumbeschreibung enthaltenen Merkmalen unterscheiden sich Räume oft nur durch ihre Ausstattung und ihre Lage im Gebäude – beides Eigenschaften, die einem Raum nicht inhärent sind – oder in ihrem Grundriss, von dem der virtuelle Raum jedoch abstrahiert. Zumeist mag der in *RoomImpl* realisierte Standard-Raum also genügen und black-box-artig verwendet werden.

Werden spezielle Räume benötigt, so muss die RoomImpl-Klasse stattdessen white-box-artig verwendet und entsprechend erweitert werden.
Spezielle Räume verlangen u.U. eine spezielle Handhabung (z.B. abschließbare Räume). Dies erfordert voraussichtlich, auch die Aspektschnittstelle Room zu erweitern und den RoomExplorer anzupassen. Das kann sehr aufwendig sein. Sinnvoll ist daher, in weiteren Entwicklungszyklen z.B. abschließbare Räume als weitere Black-Box-Komponenten vorzusehen und den RoomExplorer geeignet anzupassen.

4.1.2 Building (Gebäude)

Schnittstelle: Building
Erweiterte Schnittstelle: de.jwam.handling.thing.Thing

Implementierende Klasse: BuildingImpl (abstrakt)
Verwendung der Klasse: White-Box

Fachwert-Klasse: dvFloorPlan (erweitert dvTableOfContents)

Das fachliche Konzept des Gebäudes[385] modelliere ich in der Schnittstelle Building.
Eine abstrakte Klasse namens BuildingImpl bietet eine Basis-Implementation, die für konkrete Gebäude[386] white-box-artig verwendet werden kann.
Im Zusammenhang mit dem Gebäude wird der Fachwert dvFloorPlan neu definiert.

Die Building-Schnittstelle und die abstrakte Implementation BuildingImpl verwenden niemals direkt die Klasse RoomImpl, sondern ausschließlich die Room-Schnittstelle. Dadurch kann ein Gebäude auch mit anderen Räumen als den in RoomImpl implementierten Standard-Räumen ausgestaltet werden, sofern diese die Room-Schnittstelle erfüllen.

Konzeptionell ist ein Gebäude ein Behälter, da es Räume enthält. Der Umgang mit einem Gebäude entspricht jedoch nicht dem typischen Umgang mit einem Behälter. Üblicherweise werden Räume weder dem Gebäude hinzugefügt noch aus ihm entfernt. Demzufolge enthält die Building-Schnittstelle keine Methoden zum Hinzufügen oder Entfernen von Räumen.

[385] Kap.3.2.3.6, S.86
[386] concrete buildings ;-)

Davon abgesehen erinnert die Building-Schnittstelle sehr an die Container-Schnittstelle:

```
    public int size ();

    public boolean has (dvIdentificator id);
    public Room room (dvIdentificator id);

    public dvFloorPlan floorPlan ();
```

Schnittstelle 6: Building – Ausschnitt: Erweiterungen gegenüber Thing

Die Anzahl der enthaltenen Räume kann mit der Methode size ermittelt werden. Mit der Funktion has kann geprüft werden, ob ein bestimmter Raum Teil des Gebäudes ist. Einen bestimmten Raum erhält man über die Funktion room. Der gesamte Gebäudeplan kann schließlich über die Methode floorPlan abgefragt werden.

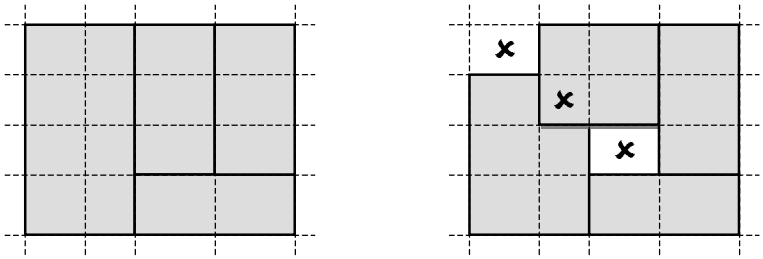
Der Gebäudeplan ist ein erweitertes Inhaltsverzeichnis eines Gebäudes – die Fachwert-Klasse dvFloorPlan spezialisiert dementsprechend den Fachwert dvTableOfContents. Neben den Raumbeschreibungen kann für jeden Raum seine Lage im Gebäude sondiert werden[387]. Auf diese Weise beschreibt der Gebäudeplan die Topologie des Gebäudes.

```
    public Rectangle rectangle (int index)
    public Rectangle rectangle (dvIdentificator id)

    public dvRoomDescription roomDescription (int index)
    public dvRoomDescription roomDescription (dvIdentificator id)

    public boolean invalidTopology ();
```

Schnittstelle 7: dvFloorPlan – Ausschnitt: Erweiterungen gegenüber dvTableOfContents

Grundlage des Gebäudeplans ist wiederum – wie bei der Anordnung von Dingen auf der Fußbodenfläche eines Raumes – ein gedachtes Raster. Räume erstrecken sich über ein Rechteck von Zellen dieses Rasters. In der folgenden Abbildung hat z.B. der rechte untere Raum im linken Raumplan die Koordinaten (2, 3, 2, 1) – (X×Y×Breite×Höhe).

[387] Wie bereits in der Schnittstelle des dvTableOfContents-Fachwerts vorgesehen, können die Einträge des Gebäudeplans wahlweise über einen Index oder den Identifikator bestimmt und sondiert werden.



gültige Topologie ungültige Topologie

Abbildung 31: Gedachtes Raster im Raumplan eines Gebäudes

Damit sich alle Räume gut sichtbar in einem GUI-Frame darstellen lassen – so wie im Entwurf des Benutzungsmodells vorgesehen[388] – dürfen sie sich nicht überlappen. Darüberhinaus sollten zwischen ihnen keine Freiräume entstehen und der gesamte Gebäudegrundriss eine rechteckige Form aufweisen. In der obigen Abbildung sind im rechten Raumplan die Zellen markiert, die dessen Topologie ungültig machen.
Nur Topologien, die diese drei Anforderungen erfüllen, werden als gültig angesehen und – dies sei im Vorgriff erwähnt – vom RoomExplorer-Werkzeug akzeptiert und angezeigt. Für die Gebäudepläne aller anderen gilt: `invalidTopology() == true`.

Die Topologie eines Gebäudes wird bei der Konstruktion des Gebäudes festgelegt.
Um während der Konstruktionsphase Räume in das Gebäude einfügen zu können, stellt die abstrakte Klasse BuildngImpl eine geschützte Methode[389] add zur Verfügung.

```
protected void add (Room r, Rectangle fpr);

protected abstract void doConstruct ();
```

Schnittstelle 8: BuildingImpl – Ausschnitt: Konstruktion eines konkreten Gebäudes

Diese Methode wird von konkreten Unterklassen – den konkreten Gebäuden – bei der Konstruktion verwendet. Dazu überschreiben sie die einzige abstrakte Methode der BuildingImpl-Klasse, doConstruct. Diese Methode wird von der BuildingImpl-Klasse im Konstruktor aufgerufen.
Da ein Gebäude ohne wenigstens einen Raum schwer vorstellbar ist, wird für doConstruct und für den BuildingImpl-Konstruktor jeweils als Nachbedingung spezifiziert, dass das konstruierte Gebäude mindestens einen Raum enthält.

Die abstrakte BuildingImpl-Klasse lässt sich nur white-box-artig verwenden. Darüberhinaus erscheint mir z.B. eine konkrete, black-box-artig verwendbare Implementation eines Gebäudes mit nur einem Raum sinnvoll. Für die Unterstützung von kleineren, kooperativ

[388] Kap.3.3.1, Abb.15, S.90
[389] Auf geschützte Methoden können in Java nur Klassen desselben Packages oder Subklassen zugreifen ([Flanagan98], S.78). Die Klassen der Anwendungsschicht eines JWAM-Systems, befinden sich prinzipiell nicht in denselben Packages, wie die Framework-Klassen aus den übrigen JWAM-Schichten. Somit können bei der Anwendungsentwicklung nur Subklassen auf geschützte Methoden des JWAM-Frameworks zugreifen.

arbeitenden Gruppen würde eine solche gemeinsame "Hütte" vermutlich in vielen Fällen genügen.

4.2 Gruppenwerkzeug RoomExplorer

Das Gruppenwerkzeug RoomExplorer soll den Benutzern eines Systems den synchronen Zugriff auf die Räumlichkeiten der gemeinsamen Umgebung, das Gebäude, ermöglichen. Das Gebäude mit seinen Räumen und den darin enthaltenen Ressourcen ist das Material bzw. Materialgeflecht, auf dem die Benutzer mit dem Gruppenwerkzeug RoomExplorer gemeinsam arbeiten. Durch die Anzeige von Awarenessinformationen macht das Gruppenwerkzeug den Benutzern ihre gemeinsamen Aktivitäten transparent. Die fachlichen Anforderungen wurden bereits detailliert im Entwurf des Benutzungsmodells besprochen[390].

Bevor ich die Komponenten des RoomExplorers im Detail erläutere, möchte ich im Folgenden einen Überblick über den Aufbau dieses Gruppenwerkzeugs geben und die grobe Funktionsweise der Teilkomponenten beschreiben.

Das Gruppenwerkzeug RoomExplorer besteht aus mehreren Komponenten. Damit ein Benutzer mit dem RoomExplorer arbeiten kann, verfügt dieser sinnvollerweise über eine Komponente, die dem Benutzer innerhalb seiner privaten Umgebung zur Verfügung steht. Gleichzeitig muss der RoomExplorer aber auch über eine allen Benutzern gemeinsame Komponente verfügen, die die Zugriffe der einzelnen Benutzer koordiniert und awarenessrelevante Änderungen am Zustand des Gebäudes an die benutzerbezogenen Komponenten meldet.

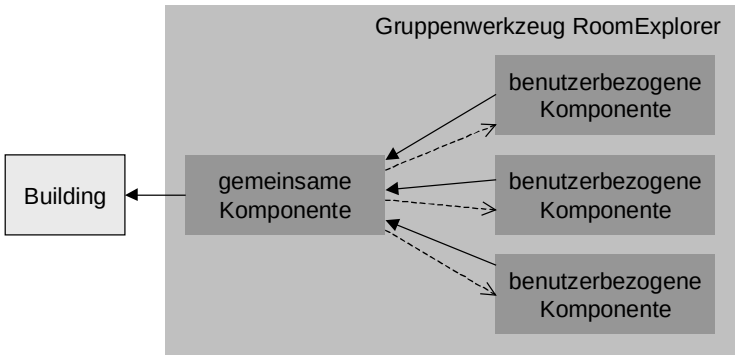


Abbildung 32: Gemeinsame und benutzerbezogene Komponenten des RoomExplorers

Die gemeinsame Komponente ist – mit den WAM-Entwurfsmetaphern gesprochen – ein Automat. Sie implementiert das synchrone Kooperationsmodell, welches Teil des Benutzungsmodells des RoomExplorers ist. Wichtige Elemente des Kooperationsmodells sind der fokussierte Raum und – vor allem – die markierte Ressource eines Benutzers[391]. Das Fokussieren ist – wie schon diskutiert – eine Einstellung am RoomExplorer, ebenso das

[390] Kap.3.3, S.89
[391] Kap.3.3.2, S.94 zum Fokussieren eines Raumes, Kap.3.3.4, S.97 zum Reservieren einer Ressource durch Markieren

Markieren. Beide Einstellungen werden von der gemeinsamen Komponente des RoomExplorers verwaltet.

An ihrer Schnittstelle stehen dafür die entsprechenden Operationen zur Verfügung. Daneben bietet die Schnittstelle der gemeinsamen Komponente Operationen zum Sondieren des Gebäudes und zum Manipulieren desselben. Letztere werden beim Aufruf über einen speziellen Fachwert "signiert", der den Benutzer repräsentiert, der den Aufruf der Operation initiiert hat.

Die benutzerbezogenen Komponenten des RoomExplorers verwenden diese Schnittstelle der gemeinsamen Komponente. Die gemeinsame Komponente versendet ihrerseits Nachrichten (Messages, Abk.: msg) an alle benutzerbezogenen Komponenten, wenn ihr Zustand verändert wurde. Der Mechanismus zum Versenden von Nachrichten ist in JWAM so realisiert, dass die Empfänger der Nachrichten dem Absender nicht bekannt sind. Die gemeinsame Komponente kennt die benutzerbezogenen Komponenten also nicht.

Es bietet sich an, die gemeinsame Komponente auf einen Server und die benutzerbezogenen Komponenten auf die lokalen Rechner der Benutzer, die Clients, zu verteilen.
Über das Proxy-Muster[392] besitzt die gemeinsame Komponente jeweils einen lokalen Stellvertreter in den verschiedenen Client-Prozessen. Die gemeinsame Komponente und ihre Proxies bieten dieselbe Schnittstelle an. Eine benutzerbezogene Komponente arbeitet jeweils auf dem Proxy, der sich in ihrem Client-Prozess befindet. Die gemeinsame Komponente auf dem Server nenne ich *RoomExplorerServer*, den Proxy *RoomExplorerServerProxy*.

Die benutzerbezogene Client-Komponente hält den über den Proxy sondierten Zustand der gemeinsamen Komponente lokal vor. Dies tut sie zum einen, um über einen konsistenten Zustand zu verfügen, zum anderen, um nicht häufiger als notwendig via Proxy auf die entfernte Komponente zugreifen zu müssen.
Damit existiert ein weiterer, clientseitiger Automat, der den am Automaten auf der Serverseite sondierten Zustand "puffert". Dieser Teil der benutzerbezogenen Komponente trägt den Namen *RoomExplorerClient*.

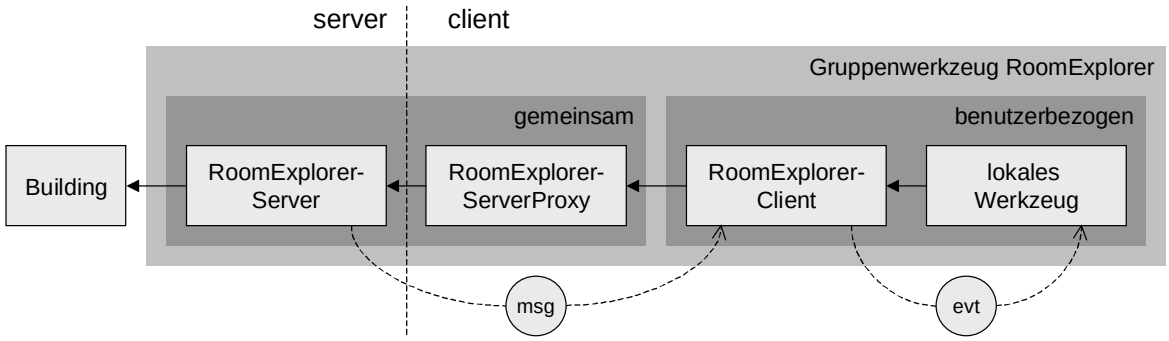


Abbildung 33: Client-, Server- und lokale Werkzeugkomponenten des RoomExplorers

[392] [Gamma97], S.227, genauer: Remote-Proxy, S.229.

Um das Gruppenwerkzeug RoomExplorer überhaupt handhaben zu können, benötigt der Benutzer ein lokales Werkzeug. Die benutzerbezogene Komponente auf dem Client besteht also aus einem RoomExplorerClient-Automaten und einem oder mehreren lokalen Werkzeugen, welche darauf arbeiten.

Die Komponenten RoomExplorerServer und RoomExplorerClient sind jeweils über eine Schnittstelle spezifiziert. Die RoomExplorerServer-Schnittstelle ist das incoming interface der RoomExplorerServer-Komponente und das outgoing interface[393] der RoomExplorerClient-Komponente. Letztere verwendet die RoomExplorerServer-Komponente nie direkt, sondern ausschließlich über diese Schnittstelle.

Auf diese Weise können verschiedene Implementationen der RoomExplorerServer-Schnittstelle black-box-artig in die RoomExplorerClient-Komponente eingebunden werden.
Der RoomExplorerServerProxy implementiert die RoomExplorerServer-Schnittstelle. Abhängig von der verwendeten Middleware können also mehrere Implementationen des Proxy angeboten und verwendet werden (z.B. ein RMI-Proxy, den ich als Standard-Proxy implementiert habe). Darüberhinaus ist auch eine nicht-verteilte Architektur denkbar, in der die RoomExplorerServer-Komponente direkt, d.h. ohne Verwendung eines Proxy, in die RoomExlorerClient-Komponente eingebunden wird – z.B. zur Unterstützung eines Gruppenarbeitsplatzes oder zur Erweiterung des Einzelarbeitsplatzes um private Räumlichkeiten.

Gegen das incoming interface der RoomExplorerClient-Komponente können die lokalen Werkzeuge programmiert werden. Für diese Werkzeuge kapselt der RoomExplorerClient die Anbindung an die gemeinsame Komponente auf dem Server. Die Werkzeuge sind dadurch von der Problematik befreit, sich mit dem Zustand der Serverkomponente zu synchronisieren. Stattdessen benachrichtigt der RoomExplorer lokale Werkzeuge über Ereignisse (Events, Abk. evt), wenn sich ein bestimmter Teil des Zustands der Serverkomponente geändert hat. Der Ereignis-Mechanismus in JWAM ist eine Implementation des Beobachter-Musters[394]. Die lokalen Werkzeuge sind der RoomExplorerClient-Komponente also nicht bekannt.

Wird die Implementation der RoomExplorerClient-Komponente verändert, z.B. um einen neuen Mechanismus zum Synchronisieren der Clients zu realisieren, kann diese neue Version der RoomExplorerClient-Komponente einfach gegen die alte ausgetauscht werden. Voraussetzung ist dabei, dass die neue Komponente das outgoing interface der lokalen Werkzeuge, also die RoomExplorerClient-Schnittstelle, implementiert.

Für alle Komponenten des Gruppenwerkzeugs RoomExplorer ist also eine black-box-artige Verwendung vorgesehen. Im folgenden beschreibe ich die Server, -Client und lokalen Werkzeugkomponenten im Detail.

---

[393] Kap.1.4.2, S.46 – die Operationen des outgoing interface einer Komponente werden von dieser an einer anderen Komponente aufgerufen, das incoming interface besteht aus den Operationen, die sie selber anbietet
[394] [Gamma97], S.257

4.2.1 RoomExplorerServer

Schnittstelle: RoomExplorerServer
Erweiterte Schnittstellen: -

Implementierende Klassen:
RoomExplorerServerImpl
RoomExplorerServerImplRMIProxy (beide implementieren RoomExplorerServer)
Verwendung der Klasse: Black-Box

Hilfsklassen:
msgRoomExplorerSnapshotChanged (erweitert de.jwam.technology.messagebroker.msgObject)
excRoomExplorer (erweitert java.lang.Exception)

Die Klasse RoomExplorerServerImpl bietet eine Standardimplementation für den serverseitigen Teil der gemeinsamen Komponente des RoomExplorers.
Die Klasse RoomExplorerServerImplRMIProxy realisiert den clientseitigen Teil unter Verwendung der Middleware RMI[395].
RoomExplorerServerImplRMIProxy ist – gemäß Proxy-Entwurfsmuster – ein lokaler Stellvertreter von RoomExplorerServerImpl. Die Schnittstelle RoomExplorerServer ist incoming interface für beide Komponenten.
Sowohl RoomExplorerServerImpl als auch der entsprechende Proxy wurden nach dem Singleton-Entwurfsmuster[396] realisiert. Im Singleton-Muster existiert von einer Klasse genau ein Exemplar, auf das über eine spezielle Klassenmethode zugegriffen wird. Auf diese Weise steht innerhalb des jeweiligen Prozesses, in dem die Komponenten verwendet werden, ein globaler Zugriffspunkt auf diese bereit.

Fast alle Operationen der Komponente RoomExplorerServerImpl werden von mehreren Threads genutzt und stellen kritische Abschnitte dar[397]. Diese müssen vor einem simultanen Zugriff geschützt werden, um auszuschließen, dass der Server in einen inkonsistenten Zustand gerät.
In Java wird dies durch die Angabe des Schlüsselworts synchronized vor einer Methodenimplementation erreicht[398]. Während des Aufrufs einer als synchronized deklarierten Methode wird das entsprechende Objekt – bzw. bei Klassenmethoden die Klasse – automatisch gesperrt[399].

---

[395] Über RMI (Remote Method Invocation) können über lokale Stellvertreter an entfernten Objekten Methoden aufgerufen werden. Die lokalen Stellvertreter werden über einen speziellen Naming-Service bezogen, den RMI anbietet. Die Verwendung von RMI ist sehr einfach und soll hier nicht weiter vertieft werden.
[396] [Flanagan98], S.139
[397] einzige Ausnahme ist der Konstruktor, der ausschließlich beim ersten Aufruf der speziellen Klassenmethode des Singleton-Entwurfsmusters aufgerufen wird – die Klassenmethode selbst ist jedoch ein kritischer Abschnitt, da sie von mehreren Threads verwendet wird
[398] z.B. `public synchronized void myMethod() { //kritischer Abschnitt }`
[399] [Flanagan98], S.41

Die Operationen der RoomExplorerServer-Schnittstelle lassen sich in drei Gruppen aufteilen. Zum einen sind dort die Operationen, über die das Gebäude im Namen eines Benutzers manipuliert werden kann. Zum anderen bietet die Schnittstelle Methoden an, mit denen der fokussierte Raum und die markierte Ressource eines jeden Benutzers explizit eingestellt werden können – implizit werden sie auch über die Methoden der ersten Gruppe beeinflusst. Die dritte Gruppe bilden schließlich die Sondierungen, die hier in Form eines Schnappschusses vorgenommen werden können.

Die Methoden der ersten beiden Gruppen verändern diesen Schnappschuss. Um die RoomExplorerClient-Komponenten davon in Kenntnis zu setzen, verschicken diese Methoden – nachdem sie erfolgreich die jeweiligen Veränderungen vorgenommen haben – die Nachricht msgRoomExplorerSnapshotChanged über den JWAM-MessageBroker[400].

Allen Methoden der Schnittstelle ist gemeinsam, dass sie eine excRoomExplorer-Ausnahme (Exception[401], Abk. exc) werfen können[402]. Die serverseitige Implementation RoomExplorerServerImpl wird dies zwar niemals tun, jedoch der clientseitige Proxy bei Verbindungsproblemen. Da beide Klassen dieselbe Schnittstelle implementieren, müssen die Ausnahmen bereits in dieser Schnittstelle deklariert werden.
Die Klasse excRoomExplorer erweitert die Java-Klasse Exception.

```
public boolean canAdd (dvUser user, dvIdentificator roomID, Thing thing);
public boolean canCarryTo (dvUser user, dvIdentificator roomID);
public boolean canRemove (dvUser user);

public boolean canAddToInventory (dvUser user);
public boolean canRemoveFromInventory (dvUser user);

public void add (
  dvUser user, dvIdentificator roomID, Point position, Thing thing);
public void carryTo (dvUser user, dvIdentificator roomID, Point position);
public void remove (dvUser user);

public void addToInventory (dvUser user);
public void removeFromInventory (dvUser user);
```

Schnittstelle 9: RoomExplorerServer – Ausschnitt: Manipulieren des Gebäudes

Alle Manipulationen am Gebäude, die über die Operationen der RoomExplorerServer-Schnittstelle vorgenommen werden, geschehen im Namen eines Benutzers. Daher wird die Operation über einen Fachwert-Parameter, der diesen Benutzer repräsentiert, "signiert".

---

[400] Der MessageBroker ist ein zentraler Nachrichtenvermittler, über den Nachrichten asynchron über Prozess- und Rechnergrenzen hinaus verschickt werden können (ausführliche Beschreibung bei [Fricke98]).
Empfänger melden sich für bestimmte Nachrichten-Klassen am MessageBroker an. Sie erfüllen eine spezielle Schnittstelle, die der MessageBroker für ihre Benachrichtigung verwendet.

[401] Eine Exception signalisiert eine abnorme Bedingung, die besonders behandelt werden muss, um einen Programmabbruch zu verhindern ([Flanagan98], S.471).

[402] in den folgenden Code-Ausschnitten fehlt hinter den Methodensignaturen aus Platzgründen und Gründen der Übersichtlichkeit vor dem Semikolon am Ende einer Zeile jeweils ... `throws excRoomExplorer`

Die Aktivitäten der RoomExplorerServer-Komponente werden grundsätzlich von einem der Benutzer angestoßen. Auch wenn sie automatisch Änderungen am Zustand des Gebäudes oder der Benutzereinstellungen an die RoomExplorerClient-Komponente meldet, wird sie doch niemals aus sich heraus aktiv, sondern erst als Folge einer Benutzeraktion.

Die meisten Methoden zum Manipulieren des Gebäudes korrespondieren jeweils mit Methoden der Room-Schnittstelle zum Hinzufügen und Entfernen von Dingen und zum Bearbeiten des Inventars[403].
Eine Ausnahme ist die Methode carryTo, mit der Dinge innerhalb eines Raumes bewegt oder von Raum zu Raum getragen werden können. Im Gegensatz zur Methode move der Room-Schnittstelle funktioniert carryTo auch raumübergreifend. Darüberhinaus besitzt carryTo eine etwas andere Semantik, nämlich das Tragen von Dingen und nicht nur das bloße Bewegen. Nach einem Aufruf von carryTo ist der dort als Parameter angegebene Raum der fokussierte Raum des Benutzers.
Gleiches gilt für die Methode add. Fügt ein Benutzer ein Ding in einen Raum ein, so fokussiert er diesen damit gleichzeitig.

Auf dem ersten Blick scheinen bei einigen Methoden einige wichtige Parameter zu fehlen. So wäre zu erwarten, dass z.B. bei der Methode remove das zu entfernende Ding und der Raum, aus dem es entfernt werden soll, angegeben werden. Dies ist nicht notwendig, da es sich um den fokussierten Raum und das markierte Ding des angegebenen Benutzers handelt. Für die Operationen add und carryTo bezeichnet der angegebene Raum den Zielraum. Der fokussierte Raum ist im Falle von carryTo der Herkunftsraum.

Die can...-Tests sind jeweils Vorbedingungen für die (fast) gleichnamigen Operationen. Der Test canRemove prüft z.B., ob der angegebene Benutzer einen Raum fokussiert und ein Ding darin ausgewählt hat.

```
public void visit (dvUser user, dvIdentificator roomID);
public void leave (dvUser user);

public void select(dvUser user, dvIdentificator thingID);
public void deselect (dvUser user);

public Thing selectedThing (dvUser user);
```

Schnittstelle 10: RoomExplorerServer – Ausschnitt: Benutzereinstellungen

Über die Methode visit kann ein Raum explizit fokussiert werden. Um einen anderen Raum zu fokussieren, muss der bisher fokussierte Raum nicht extra mit der Methode leave "verlassen" werden. Dies geschieht implizit beim Aufruf von visit.
Desgleichen muss eine Ressource nicht erst mit deselect freigegeben werden, bevor eine andere mit select markiert werden kann.

[403] Kap.4.1.1, Schnittstelle 1, S.122 und Schnittstelle 2, S.123

Mit der Funktion selectedThing kann auf das von einem Benutzer ausgewählte Ding zugegriffen werden. Diese Sondierung ist nicht Teil des Schnappschusses (s.u.), da dieser lediglich Fachwerte enthält. Verwendet wird selectedThing, um ein Ding technisch auf den Client zu kopieren, z.B. wenn ein Benutzer das Ding aus dem Gebäude heraus auf den Einzelarbeitsplatz bewegen will.

Die Funktion selectedThing entfernt das Ding nicht aus seinem Raum – sie hat also keinen Nebeneffekt. Um ein Ding vollständig auf den Client zu bewegen, muss es mit der Methode remove explizit aus seinem Raum entfernt werden, nachdem es mit selectedThing kopiert wurde. Sollte remove eine Exception werfen, so ist das Bewegen des Dinges fehlgeschlagen. Obwohl sich bereits eine technische Kopie des Dinges auf dem Client befindet, darf diese dann nicht verwendet werden. Andernfalls wäre die Einheit von Raum und Zeit durchbrochen.

```
public RoomExplorerSnapshot snapshot ();

public RoomExplorerSnapshot.Changes changes (
  RoomExplorerSnapshot.Counter outdatedCounter);
```

Schnittstelle 11: RoomExplorerServer – Ausschnitt: Sondieren des Schnappschusses

Ein Schnappschuss des Zustands des RoomExplorerServers, d.h. des Zustands des dahinter verborgenen Gebäudes und der Benutzereinstellungen am RoomExplorerServer, kann über die Funktion snapshot ermittelt werden.
Wurde zu einem früheren Zeitpunkt bereits ein Schnappschuss sondiert, so liefert die Funktion changes die veränderten Teile des aktuellen Schnappschusses gegenüber dem bereits sondierten.
Um dies verständlicher zu machen, erläutere ich im folgenden Unterkapitel das Schnappschuss-Konzept im Detail.

4.2.2 Ein Schnappschuss des Gebäudezustands: RoomExplorerSnapshot

Schnittstellen:
RoomExplorerSnapshot
RoomExplorerSnapshot.Changes
RoomExplorerSnapshot.Counter
Erweiterte Schnittstellen: -

Implementierende Klassen:
RoomExplorerSnapshotImpl
RoomExplorerSnapshotImpl.ChangesImpl
RoomExplorerSnapshotImpl.CounterImpl
Verwendung der Klassen: Black-Box

Die Schnittstelle RoomExplorerSnapshot definiert einen Schnappschuss des RoomExplorerServer-Zustands. RoomExplorerSnapshotImpl ist die Standard-Implementierung.
Für den Satz an Änderungen und die laufende Nummer verwende ich sogenannte *innere Schnittstellen* und *Klassen*[404] – daher die besondere Schreibweise von RoomExplorerSnapshot.Changes und RoomExplorerSnapshot.Counter. Auf diese Weise drücke ich aus, dass diese Konzepte Bestandteil des Schnappschuss-Konzepts sind, ohne das sie keinen eigenständigen Wert hätten.

Aufgrund des synchronen Benutzungsmodells ist es für die RoomExplorerClient-Komponente problematisch, den Zustand des RoomExplorerServers zu sondieren.
Sondiert der RoomExplorerClient z.B. zunächst das Positionenverzeichnis eines Raumes, in dem ein anderer Benutzer eine Ressource markiert hat, ist leicht vorstellbar, dass dieser Benutzer die Ressource sofort danach in einen anderen Raum verschiebt. Sondiert der RoomExplorerClient nun das Positionenverzeichnis des anderen Raumes, so befindet sich das verschobene Ding scheinbar in beiden Räumen gleichzeitig. Der sondierte Gesamtzustand wäre somit nicht konsistent.
Die Konsistenz des sondierten Zustands könnte sichergestellt gestellt werden, indem der Zustand entweder auf einmal sondiert wird oder zunächst "eingefroren" und dann in Schritten sondiert wird.

Letzteres hieße, den gesamten Sondierungsvorgang eines RoomExplorerClients als kritischen Abschnitt zu betrachten und währenddessen den RoomExplorerServer gegen Änderungen zu sperren. Dies wäre in Anbetracht des synchronen Benutzungsmodells keine gute Lösung. Die entstehenden Wartezeiten wären sehr wahrscheinlich unvertretbar lang. Gerade nach einer Änderung, wenn alle Clients ihren Zustand mit dem Server synchronisieren müssen, käme es vermutlich zu Engpässen.

Da die Clients den Serverzustand ohnehin lokal vorhalten müssen, bietet es sich an, diesen in einer Art Schnappschuss bereits am RoomExplorerServer bereitzustellen. Der Serverzustand könnte als Schnappschuss im Ganzen – und somit auf jeden Fall konsistent – sondiert werden. Für diese Lösung habe ich mich entschieden.

Ein Schnappschuss des Serverzustands kann sehr umfangreich sein, wie ein Blick auf die Schnittstelle zeigt:

[404] Innere Klassen werden als Elemente anderer Klassen definiert – genau wie z.B. Methoden und Felder. In diesem Fall handelt es sich um sog. Top-Level-Klassen, die sich wie "normale" Klassen verhalten. Hier zeigt sich die Schachtelung lediglich im Namen. Gleiches gilt für die inneren Schnittstellen (siehe [Flanagan98], S.110f.).

```
    public boolean hasSelected (dvUser user);
    public dvIdentificator selectedThingID (dvUser user);

    public boolean isVisiting (dvUser user);
    public dvIdentificator visitedRoomID (dvUser user);

    public dvTableOfPositions tableOfPositions (dvIdentificator roomID);
    public dvTableOfContents inventoryList (dvIdentificator roomID);

    public dvFloorPlan floorPlan ();

    public dvUser[] existingUsers ();
```

Schnittstelle 12: RoomExplorerSnapshot – Ausschnitt: Sondieren des Serverzustands

Er enthält für jeden Raum dessen Positionentabelle – die an sich schon recht umfangreich sein kann – und die Inventarliste, die jeweils über die Funktionen tableOfPositions und inventoryList sondiert werden können.
Der Gebäudeplan kann über die Methode floorPlan ermittelt werden.
Zudem enthält ein Schnappschuss für jeden Benutzer den Identifikator des "besuchten", d.h. fokussierten Raumes und den Identifikator des von ihm markierten Dinges. Diese können mit visitedRoomID und selectedThingID bestimmt werden. Über die Funktionen isVisiting und hasSelected kann festgestellt werden, ob ein bestimmter Benutzer einen Raum fokussiert, bzw. ein Ding markiert hat.
Schließlich kann über existingUsers ein Array aller existierenden Benutzer erfragt werden. Diese Information benötigt die benutzerbezogene Komponente des RoomExplorers, damit sie für diese Benutzer deren Benutzereinstellungen, d.h. den Raumfokus und das markierte Ding, sondieren und als Awarenessinformationen visualisieren kann.

Nachdem ein Client einen Schnappschuss sondiert hat, ist dieser Schnappschuss potentiell schon veraltet. Der Serverzustand könnte sich durch die Aktion eines anderen Benutzers bereits geändert haben.

Der Serverzustand ändert sich meist nur in wenigen Teilen, nämlich in den Benutzereinstellungen und der Position bzw. dem Platz der markierten Ressource eines Benutzers. Der Gebäudeplan ändert sich z.B. nie. Es ist für die Clients daher nicht notwendig, nach jeder Änderung den gesamten Schnappschuss zu aktualisieren. Effizienter ist, nur die Änderungen zwischen zwei Schnappschüssen am entfernen RoomExplorerServer zu sondieren.

Schnappschüsse werden, wenn sie von der RoomExplorerServer-Komponente erstellt werden, laufend durchnumeriert. Die Schnittstelle RoomExplorerSnapshot.Counter spezifiziert eine solche laufende Nummer.

Die Schnittstelle RoomExplorerSnapshot.Changes definiert einen Satz von Änderungen, die ein Schnappschuss gegenüber einem älteren widerspiegelt.
Die laufenden Nummern beider Schnappschüsse können an dem Satz von Änderungen sondiert werden – auf diese Weise kann sichergestellt werden, dass nicht versehentlich Änderungen ausgelassen werden, wenn die Schnappschüsse aktualisiert werden.

Im übrigen ist die RoomExplorerSnapshot.Changes-Schnittstelle sehr generisch: Es werden einfach neue Werte zusammen mit einer laufenden Nummer unter einem Schlüsselwert abgelegt. Unter demselben Schlüsselwert können die Änderungen auch wieder abgefragt werden. Die Wahl der konkreten Schlüsselwerte liegt bei der Implementation des RoomExplorerSnapshot.

Um nur die tatsächlichen Änderungen austauschen zu müssen, bietet der RoomExplorerSnapshot die folgenden Operationen an:

```
    public Counter counter ();

    public Changes changes (Counter outdatedCounter);
    public Changes newChanges ();

    public RoomExplorerSnapshot updatedCopy (Changes changes);
```

Schnittstelle 13: RoomExplorerSnapshot – Ausschnitt: Handhaben von Änderungen

Die Methode counter liefert die laufende Nummer eines Schnappschusses.
Mit der Methode changes können die Änderungen an einem Schnappschuss gegenüber dem zuletzt sondierten Schnappschuss ermittelt werden. Dessen laufende Nummer wird der Funktion als Parameter übergeben.
Die Funktion newChanges liefert einen neuen, leeren Satz an Änderungen zurück. Die laufende Nummer dieser Änderungen ist um eins höher als die des RoomExplorerSnapshot-Objekts, an dem newChanges aufgerufen wurde. Dieser wird vom RoomExplorerServer benötigt, um beim Aufruf einer verändernden Operation die vollzogenen Änderungen darin zu protokollieren. Anschließend aktualisiert der RoomExplorerServer damit den bisher gültigen Schnappschuss.
Dazu verwendet er die Methode updatedCopy. Wird sie an einem RoomExplorerSnapshot-Objekt aufgerufen, so liefert sie eine aktualisierte Kopie zurück, in die der übergebene Satz an Änderungen übernommen wurde. Die laufende Nummer der aktualisierten Kopie entspricht der aktuelleren der beiden laufenden Nummern des übergebenen Änderungssatzes.
Da ein Schnappschuss praktisch nur aus Fachwerten besteht, die nach dem Fliegengewicht-Entwurfsmuster[405] realisiert werden, muss beim Erzeugen dieser Kopie kaum etwas tatsächlich kopiert werden. Bzgl. Speicherbedarf und Laufzeitverhalten ist dieses Verfahren also recht effizient.

[405] [Gamma97], S.199 – durch dieses Muster besteht von jedem Fachwert-Objekt nur ein einziges Exemplar

Der RoomExplorerClient kann die Methode changes nicht an dem aktuellen RoomExplorerSnapshot-Objekt des RoomExplorerServers aufrufen, da es nicht wie dieser über einen lokalen Stellvertreter verfügt. Aus diesem Grund bietet der RoomExplorerServer ebenfalls eine Methode changes an (s.o.). Diese ruft ihrerseits die Methode changes am aktuellen Schnappschuss des Serverzustands auf.

4.2.3 RoomExplorerClient

Schnittstelle: RoomExplorerClient
Erweiterte Schnittstelle: de.jwam.technology.messagebroker.MessageRecipient

Implementierende Klasse: RoomExplorerClientImpl
Verwendung der Klasse: Black-Box

Die Schnittstelle RoomExplorerClient definiert den clientseitigen, benutzerbezogenen Automaten des RoomExplorer-Gruppenwerkzeugs. Auch hier existiert mit der Klasse RoomExplorerClientImpl eine Standardimplementation der Schnittstelle. Diese Klasse wurde – wie schon RoomExplorerServerImpl und der entsprechende Proxy – nach dem Singleton-Entwurfsmuster realisiert.

Da die Schnittstelle des RoomExplorerClients nur von einem Benutzer verwendet wird, kann sie einfacher sein, als die Schnittstelle des RoomExplorerServers. Die Operationen müssen nicht extra "signiert" werden, da sie grundsätzlich im Namen des Benutzers ausgeführt werden, in dessen privater Umgebung sich die benutzerbezogene Komponente des RoomExplorers befindet.

Im übrigen gleicht die Schnittstelle sehr der RoomExplorerServer-Schnittstelle. Im Gegensatz zu diesen haben sie allerdings eine etwas veränderte Semantik. Wogegen auf dem Server jede Operation ein konkretes Ergebnis zusichern kann, ist dies auf dem Client nicht mehr möglich. Aufgrund von möglichen Verbindungsproblemen besteht immer die Gefahr, dass eine Operation nicht durchführbar ist. Alle Operationen der RoomExplorerClient-Schnittstelle, die sich auf den Serverzustand beziehen, stellen daher lediglich einen Versuch dar, auf den Serverzustand zuzugreifen. In der Benennung der Operationen kommt das durch den Zusatz "tryTo" bzw. "canTryTo" zum Ausdruck. Zudem werfen all diese Operationen – wie die Operationen des RoomExplorerServer-Proxy – bei Verbindungsproblemen Exceptions.

```
public boolean canTryToGetSelectedThing ();

public boolean canTryToAdd (
  dvIdentificator roomID, Point position, Thing thing);
public boolean canTryToCarryTo (dvIdentificator roomID, Point position);
public boolean canTryToRemove ();
public boolean canTryToAddToInventory ();
public boolean canTryToRemoveFromInventory ();

public Thing tryToGetSelectedThing ();

public void tryToVisit (dvIdentificator roomID);
public void tryToLeave ();
public void tryToSelect (dvIdentificator thingID);
public void tryToDeselect ();

public void tryToAdd (dvIdentificator roomID, Point position, Thing thing);
public void tryToCarryTo (dvIdentificator roomID, Point position);
public void tryToRemove ();
public void tryToAddToInventory ();
public void tryToRemoveFromInventory ();
```

Schnittstelle 14: RoomExplorerClient – Ausschnitt: "tryTo"- und "canTryTo"-Operationen

Mit den "canTryTo"-Operationen kann getestet werden, ob es überhaupt sinnvoll ist, die korrespondierende "tryTo"-Operation aufzurufen. So wäre es z.B. nicht sinnvoll zu versuchen, ein Ding zu verschieben, wenn es zuvor nicht markiert wurde. Ein solcher Versuch schlüge in jedem Falle fehl.

Die RoomExplorerClient-Schnittstelle erweitert die JWAM-Schnittstelle MessageRecipient. Dies ist die Schnittstelle, unter der sich Empfänger von Nachrichten am MessageBroker anmelden.
Die RoomExplorerClient-Komponente meldet sich für die von der RoomExplorerServer-Komponente versandte Nachrichten-Klasse msgRoomExplorerSnapshotChanged an.

Erhält die RoomExplorerClient-Komponente vom MessageBroker eine solche Nachricht, ruft sie an sich selbst die Methode tryToUpdateSnapshot auf. In dieser Methode ermittelt sie an der RoomExplorerServer-Komponente (bzw. an deren Proxy) mit der Methode changes den Satz an Änderungen und erzeugt damit einen neuen Schnappschuss. Anschließend benachrichtigt sie die auf ihr arbeitenden lokalen Werkzeuge über die einzelnen Änderungen mithilfe von Events.
Tritt beim Aufruf der Methode changes ein Verbindungsproblem auf, so bricht die Methode tryToUpdateSnapshot den Versuch, den Schnappschuss zu aktualisieren, ab. Die RoomExplorerClient-Komponente und die lokalen Werkzeuge bleiben in ihrem alten Zustand – obwohl dieser, wie die Nachricht msgRoomExplorerSnapshotChanged angezeigt hat, offenbar veraltet ist.

Aufgrund des synchronen Benutzungsmodells ist der angezeigte Zustand potentiell immer veraltet, da andere Benutzer den Serverzustand verändert haben können, kaum dass ein Client seinen Zustand synchronisiert hat. Das Verbindungsproblem bedeutet zwar eine (u.U. sehr lange) Verzögerung, stellt aber kein Konsistenzproblem dar. Durch die laufende Durchnumerierung der Schnappschüsse werden die nicht ermittelbaren Änderungen beim nächsten geglückten Versuch berücksichtigt, da die laufende Nummer des nicht aktualisierten Schnappschusses unverändert geblieben ist und abermals der Methode changes übergeben wird. Früher oder später wird also der korrekte Zustand ermittelt.

```
public RoomExplorerSnapshot snapshot ();

public evtObject evtTableOfPositionsChanged (dvIdentificator roomID);
public evtObject evtInventoryListChanged (dvIdentificator roomID);
public evtObject evtVisitedRoomChanged (dvUser user);
public evtObject evtSelectedThingChanged (dvUser user);
```

Schnittstelle 15: RoomExplorerClient – Ausschnitt: Ermitteln des "gepufferten" Schnappschusses und der Events zu dessen Änderungen

Die lokalen Werkzeuge ermitteln den von der RoomExplorerClient-Komponente "gepufferten" Schnappschuss mit der Methode snapshot. Diese Methode ist im Gegensatz zu den "tryTo"-Operationen "verlässlich" und wirft keine Exceptions.
Zudem können sich die lokalen Werkzeuge an Events zu konkreten Änderungen am Snapshot anmelden. Diese Events können an der RoomExplorerClient-Schnittstelle ermittelt werden.

4.2.4 Die lokale Werkzeugkomponente: RoomExplorer und RoomTop

Klassen[406]:

toolRoomExplorer, ipRoomExplorer, fpRoomExplorer
toolRoomTop, ipRoomTop, fpRoomTop
(diese Klassen erweitern die jeweiligen Basisklassen zur Werkzeugkonstruktion im Package de.jwam.handling.toolconstruction.basicconstruction)

Die lokale Werkzeugkomponente besteht aus zwei Teilen: dem sogenannten Kontextwerkzeug RoomExplorer und dem darin (in mehreren Exemplaren) eingebetteten Subwerkzeug RoomTop. Dies wurde bereits beim Entwurf des Benutzungsmodells so vorgesehen[407].

[406] Die hier beschriebenen Klassen ergeben sich unmittelbar aus dem JWAM-Entwurfsmuster zur Werkzeugkonstruktion. Hier nicht weiter berücksichtigt sind die GUI-Klassen, zu denen auch neue Präsentations- und Interaktionsformen für ein Icon-Widget gehören, das in der Lage ist, die in Kap 3.3.5, S.98 beschriebenen Awarenessinformationen anzuzeigen. Die neue PF und IAF befinden sich in der Technologieschicht im Package de.jwamx.technology.iafpf.
[407] Kap.3.3.1, S.90

An dieser Stelle möchte ich nicht wie zuvor bei den anderen Komponenten geschehen, die Schnittstelle der Werkzeugklassen genauer betrachten. Diese ergeben sich hauptsächlich aus dem JWAM-Entwurfsmuster zur Werkzeugkonstruktion.

Die lokale Werkzeug-Komponente entspricht in ihrem Aufbau bereits dem neuen Entwurfsmuster zur Konstruktion von Werkzeugen mit JWAM[408]. Im Gegensatz zur alten Konstruktionsweise haben Kontext- und Subwerkzeuge nun die gleiche Struktur. Das ermöglicht, Werkzeuge "alleinstehend" oder als Subwerkzeuge in verschiedenen Kontextwerkzeugen zu verwenden.

Ein JWAM-Werkzeug besteht i.d.R. aus drei Teilkomponenten[409]:

- Funktionskomponente (functional part, Abk. fp): implementiert die fachliche Funktionalität des Werkzeugs
- Interaktionskomponente (interaction part, Abk. ip): implementiert die Handhabung und Präsentation des Werkzeugs
- Werkzeugabschluss (tool): konfiguriert die beiden anderen Komponenten und repräsentiert das gesamte Werkzeug nach außen.

Funktions- und Interaktionskomponente sind über den JWAM-Event-Mechanismus, eine spezielle Implementierung des Beobachter-Entwurfsmusters[410], lose gekoppelt: Die Funktionskomponente stellt an ihrer Schnittstelle Event-Objekte zur Verfügung. Die Interaktionskomponente kennt ihre Funktionskomponente und meldet sich bei den Events, die für sie relevant sind, als Beobachter an. Die Funktionskomponenten veranlassen – z.B. bei Änderungen des bearbeiteten Materials – die entsprechenden Events dazu, ihre Beobachter zu benachrichtigen, die dann ihrerseits auf das Ereignis reagieren können.

Auch die Funktionskomponenten von Kontext- und Subwerkzeugen sind über den Event-Mechanismus lose gekoppelt. Das Kontextwerkzeug beobachtet dabei seine Subwerkzeuge.

Zudem kommt dieser Mechanismus auch – wie schon erwähnt – bei der Kopplung der lokalen Werkzeuge an die RoomExplorerClient-Komponente zum Einsatz. Hierbei beobachten die Werkzeuge den RoomExplorerClient-Automaten.

Die folgende Abbildung illustriert den Aufbau des lokalen Werkzeugs und die Kopplung der Teilkomponenten:

[408] Hierzu ist mir leider keine Dokumentation bekannt, die ich als Quelle angeben könnte – abgesehen von den Kommentaren im Quelltext des Packages de.jwam.handling.toolconstruction.basicconstruction und dem Quelltext selbst. Die Werkzeugkonstruktion nach WAM im Allgemeinen wird in [WAM98], S.479ff. beschrieben.

[409] Es sind auch kompakte Werkzeuge konstruierbar, die z.B. Funktions- und Interaktionskomponente nicht trennen – sehr kleine und einfache Werkzeuge sind so bequemer zu realisieren.

[410] [Gamma97], S.257

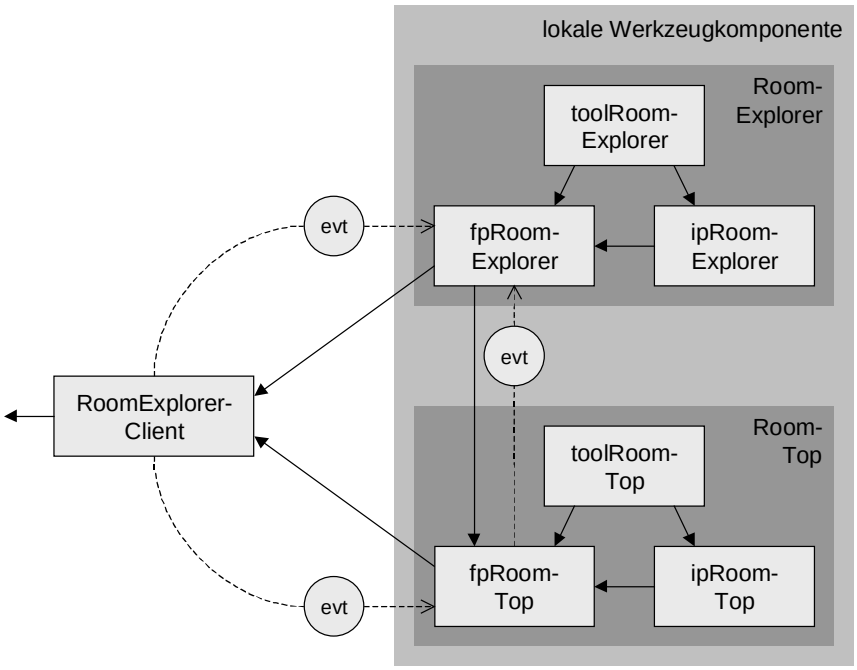


Abbildung 34: RoomExplorerClient, RoomExplorer und RoomTop

Der Aufbau der lokalen Werkzeugkomponente ergibt sich aus der Topologie des Gebäudes.

Wird das Kontextwerkzeug RoomExplorer lokal auf einem Client gestartet, sondiert es zunächst die RoomExplorerClient-Komponente.
Diese ist – wie beschrieben – ein Singleton. Beim ersten Zugriff über die entsprechende Singleton-Klassenmethode wird das einzige Exemplar dieser Komponente erzeugt. Dabei stellt sie die Verbindung zur RoomExplorerServer-Komponente her und ermittelt als erstes den Schnappschuss des Serverzustands, um ihn für die lokale Werkzeugkomponente zu "puffern".
Der lokale RoomExplorer sondiert an diesem "gepufferten" Schnappschuss den Gebäudeplan und erzeugt damit die benötigten RoomTops – für jeden Raum einen. Diese werden mit dem Identifikator des jeweiligen Raumes konfiguriert, so dass sie sich selbständig initialisieren können.
Dazu sondieren auch sie an der RoomExplorerClient-Komponente den "gepufferten" Schnappschuss und ermitteln daran Positionenverzeichnis und Inventarliste ihres Raumes, Awarenessinformationen zu allen existierenden Benutzern sowie (über den Gebäudeplan) ihre Raumbeschreibung. Damit sind die RoomTops in der Lage, ihren Raum so wie beim Entwurf des Benutzungsmodells gezeigt zu visualisieren.

Das folgende Interaktionsdiagramm illustriert diesen Ablauf[411]:

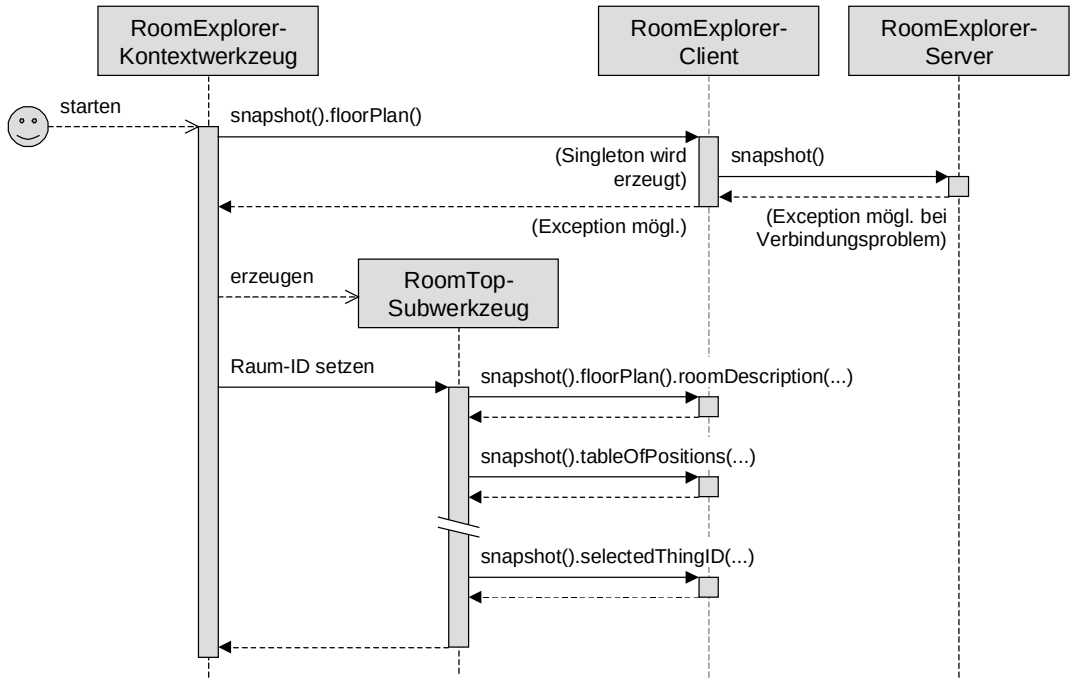


Abbildung 35: Interaktionsdiagramm zum Start der lokalen Werkzeugkomponente

Das RoomExplorer-Kontextwerkzeug ordnet die enthaltenen RoomTops gemäß der Topologie des zu visualisierenden Gebäudes an.
Sowohl das RoomExplorer-Kontextwerkzeug als auch das RoomTop-Subwerkzeug sind generisch aufgebaut. Gemeinsam sind sie in der Lage, ein beliebiges Gebäude mit einer gültigen Topologie darzustellen und für einen Benutzer handhabbar zu machen.
Ist die Topologie des Gebäudes ungültig, erzeugt das RoomExplorer-Kontextwerkzeug keine Subwerkzeuge und zeigt stattdessen einen leeren GUI-Frame an. Das Werkzeug kann vom Benutzer dann nur noch geschlossen werden und eröffnet darüberhinaus keine Interaktionsmöglichkeiten.

Bei der Handhabung des visualisierten Gebäudes realisiert das RoomTop-Subwerkzeug die wichtigsten Interaktionsmöglichkeiten – dazu gehören z.B. das Fokussieren eines Raumes, das Markieren einer Ressource, das Bewegen von Dingen usw. Lediglich das Umschalten zwischen Raum- und Gebäudeansicht liegt im Aufgabenbereich des RoomExplorer-Kontextwerkzeugs.

[411] Die Aufteilung der Werkzeuge in ihre drei Komponenten habe ich hier nicht dargestellt, um das Interaktionsdiagramm damit nicht zu überladen.

5. Zusammenfassung und Ausblick

Im Rahmen dieser Arbeit wurde eine Raumkomponente für die situative Kooperationsunterstützung entworfen und auf der Grundlage des JWAM-Frameworks in einer Basisversion implementiert.

5.1 Die Idee hinter der Raumkomponente

Vielen Softwaresystemen liegt eine räumliche Vorstellung zugrunde, so z.B. mit dem Begriff der Arbeitsumgebung des traditionellen Einzelarbeitsplatzes.
Ein Ort ist eine räumliche Umgebung um einen "Brennpunkt", um den sich Dinge und Personen gruppieren. Mit ihnen vebindet sich eine allgemein geteilte Vorstellung von Wahnehmungs-, Handlungs- und Interaktionsmöglichkeiten an diesem Ort. Darüberhinaus besitzt ein Ort eine bestimmte Struktur, mit der fachliche Bedeutung verknüpft werden kann.

Durch die an einen Ort geknüpfte fachliche Bedeutung und damit verbundenen Konventionen wird ein Ort zu einem Platz. Ein solches spezifisches Platzverständnis entwickeln Personen im Umgang mit den Orten. Wenngleich es nicht durch den Entwurf eines Systems vorgegeben werden kann, kommt es dennoch in der Gestaltung der virtuellen Plätze zum Ausdruck.
Durch die Verknüpfung von Konventionen und räumlichen Eigenschaften ermöglichen Plätze die Orientierung in einer kooperativ genutzten Umgebung. Alle Dinge haben ihren Platz im System, der von den Benutzern benannt werden kann und mit einem gemeinsamen Verständnis belegt ist.
Einzelarbeitsplätze sowie virtuelle Gebäude, Räume, Möbel und Stapel sind Beispiele für Plätze.

Der traditionelle Einzelarbeitsplatz hat als private Umgebung auch neben der gemeinsamen Umgebung, die durch die virtuellen Räumlichkeiten vergegenständlicht wird, weiterhin Bestand. Die Einzelarbeitsplätze der übrigen Benutzer sind für einen Benutzer nicht sichtbar und gehören nicht zu seiner individuellen Umgebung. Diese setzt sich aus dessen privatem Einzelarbeitsplatz und den gemeinsamen Räumlichkeiten zusammen.

Die gemeinsamen Räumlichkeiten setzen sich zu einem virtuellen Gebäude zusammen. Dieses vergegenständlicht die gesamte gemeinsame Umgebung.
Die Räume, aus denen es besteht, sind grundrissartig angeordnet. Sie sind nicht durch Türen verbunden, sondern direkt erreichbar. Wege durch das Gebäude spielen somit bei der Navigation keine Rolle – dazu genügt Überblickswissen.
In einen Raum lassen sich Dinge einfügen und entnehmen. Enthaltene Dinge werden auf der Fußbodenfläche des Raumes zweidimensional angeordnen.
Benutzer können innerhalb des gemeinsamen Gebäudes Stammplätze für bestimmte Ressourcen festlegen, indem sie diese in das Inventar eines Raumes aufnehmen.

Die gemeinsame Umgebung ist durch das Gebäude und die Räume, sowie das Inventarkonzept klar und in Analogie zur physischen Welt strukturiert.

Im Gegensatz zu den physischen Räumen gehört es allerdings nicht zum Umgang mit einem virtuellen Raum, ihn zu betreten. Ein virtueller Raum und auch andere virtuelle Plätze werden stattdessen über ein Werkzeug fokussiert.

Die Analogie zur physischen Arbeitswelt gestattet die Übernahme von bestehenden Erfahrungen, Konventionen, Sprechweisen usw., die sich in der physischen Arbeitsumgebung bewährt haben. Die Raumkomponente ist dadurch weitaus mehr, als nur eine weitere Möglichkeit zum Materialaustausch.

5.2 Die Handhabung und Präsentation der Raumkomponente

Die in dieser Arbeit entworfene Raumkoponente visualisiert das gemeinsame Gebäude. Dabei ist kaum visuelle Phantasie vonnöten, um sich ein Bild vom Aufbau der gemeinsamen Umgebung zu machen.

Die Räume des gemeinsamen Gebäudes werden im sogenannten RoomExplorer-Werkzeug grundrissartig präsentiert. Subwerkzeuge des RoomExplorers, die sogenannten RoomTops, visualisieren in diesem Grundriss jeweils einen einzelnen Raum, indem sie dessen Fußbodenfläche desktop-artig darstellen. Der "Teppichboden" ist dabei das auffälligste visuelle Erkennungsmerkmal eines Raumes.

Alle Dinge in einem Raum sind unmittelbar und für jeden gleichermaßen sichtbar. Es gilt: "What You See Is What I See" (WYSIWIS). Zudem können alle Benutzer in gleicher Weise auf die Räume und die darin enthaltenen Ressourcen zugreifen. In Analogie zu WYSIWIS gilt also auch: "What You Can Do Is What I Can Do".
Dinge können von einem Benutzer im RoomExplorer von Raum zu Raum oder innerhalb eines Raumes an eine andere freie Position verschoben werden. Zudem können sie aus dem RoomExplorer heraus auf den Desktop bewegt werden und zurück. Es existieren innerhalb der individuellen Umgebung eines Benutzers keine unsichtbaren "Barrieren", die ihn daran hindern, Dinge umherzubewegen

All diese Tätigkeiten beziehen sich jeweils auf den fokussierten Raum eines Benutzers. Wird ein Ding in einen anderen Raum verschoben, so verschiebt sich auch der Fokus.
Benutzer, die denselben Raum fokussiert haben, koordinieren sich, indem sie das Ding, auf das sie zugreifen möchten, markieren und damit für die exklusive Nutzung reservieren.
Alle Tätigkeiten am RoomExplorer lassen sich grundsätzlich mit der Maus vollziehen, z.B. via Drag&Drop. Das Markieren eines Dinges geschieht beim Drag&Drop implizit und erfordert keine zusätzliche Aktion.

Welches Ding von einem Benutzer markiert wurde, wird als Awarenessinformation an den RoomExplorern der anderen Benutzer angezeigt – ebenso welcher Raum von einem Benutzer fokussiert wird. Bestehende und "drohende" Konkurrenzsituationen können so erkannt und die eigenen Handlungen daran nach eigenem Ermessen ausgerichtet werden.

5.3 Die Konstruktion der Raumkomponente

Die Raumkomponente ist in der Handhabungs- und Präsentationsschicht des JWAM-Frameworks angesiedelt.

Über Schnittstellen werden die Hot-Spots der Basis-Raumkomponente definiert. Zu den Hot-Spots existieren Teilkomponenten, wie z.B. einem white-box-artig verwendbaren Gebäude, das spezialisiert und mit black-box-artig verwendbaren Räume ausgestattet werden muss.

Der RoomExplorer verwendet eine Client-Server-Architektur. Er besteht aus Komponenten, die den Benutzern jeweils an ihren Einzelarbeitsplätzen zur Verfügung stehen und einer gemeinsamen Komponente, die die Zugriffe koordiniert und awarenessrelevante Änderungen an die benutzerbezogenen Komponenten meldet. All diese Teilkomponenten sind black-box-artig verwendbar.

Die gemeinsame Komponente ist – nach WAM – ein Automat. Die benutzerbezogenen Komponenten bestehen jeweils aus einem weiteren Automaten, der die Anbindung an die gemeinsame Komponente kapselt, und einem lokalen Werkzeug, das darauf arbeitet.
Gegen die Schnittstelle des Automaten der benutzerbezogenen Komponente können weitere lokale Werkzeuge programmiert werden.

5.4 Ausblick

Das RoomExplorer-Werkzeug ermöglicht mehreren Benutzern, auf die Räumlichkeiten der gemeinsamen Umgebung zuzugreifen – es handelt sich somit um ein Gruppenwerkzeug. Auch der Registrar der JWAM-Registratur ist ein solches Gruppenwerkzeug.
Über weitere Gruppenwerkzeuge lässt sich die Funktionalität des RoomExplorers beträchtlich erweitern. So kann z.B. ein Gruppenwerkzeug zum Lesen von Notizen in Räumen als Subwerkzeug in den RoomExplorer eingebettet werden. Desweiteren werden Gruppenwerkzeuge benötigt, um inventarisierte Dinge direkt in den Räumen bearbeiten zu können, da diese nicht auf den Desktop verschiebbar sind. Für derartige Gruppenwerkzeuge kann ein erweiterter RoomExplorer die Möglichkeit bieten, sie z.B. per Doppelklick auf das zu bearbeitende Ding zu starten.

In der realisierten Basis-Raumkomponente sind Registraturen bisher die einzigen Möbel. Hier kann man sich weitere vorstellen, wie z.B. virtuelle Tische, die über eine desktop-artig visualisierbare Arbeitsfläche verfügen. Zudem sind virtuelle Stapel denkbar, die auf solchen Arbeitsflächen oder der Fußbodenfläche eines Raumes liegen können.

In einer erweiterten Raumkomponente sollten Räume abschließbar sein, um expliziten Zugriffsschutz zu gewähren.

Desgleichen sind abschließbare Trakte und Etagen vorstellbar. Etagen dienen zudem der Konstruktion großer virtueller Gebäude, die sich z.B. etagenweise visualisieren lassen. Große,

„systemnahe“ Automaten, wie z.B. die Postdienststelle des JWAM-Postversandsystems, lassen sich in einer „Kelleretage“ unterbringen, die z.B. nur der Systemadministrator – der „Hausmeister des Systems“ – betreten darf.

Über eine an einen Raum geknüpfte Chat- oder Audio-Verbindung kann die explizite synchrone Kommunikation zwischen Personen, die denselben Raum fokussiert haben, auf elegante Weise unterstützt werden.

Der Zugriff auf das Gebäude kann über ein Gebäudeplan-Menü direkt aus Werkzeugen heraus ermöglicht werden. Ein Web-Interface gestattet den Zugriff über einen Web-Server, ohne dass dafür – abgesehen von einem gewöhnlichen Internet-Browser – eine spezielle Software vonnöten ist.

Bei der Gestaltung eines virtuellen Gebäudes stellt sich die Frage, wie man zu geeigneten Räumen kommt. Sollte dies immer auf der Grundlage physischer Vorbilder geschehen? Möglicherweise existieren diese nicht, da gemeinsame Räumlichkeiten erst durch die Einführung des Systems zur Verfügung gestellt werden.
Evtl. hilft die Orientierung an Kooperationsbildern, in denen die für die Kooperation relevanten Plätze des Anwendungsbereichs visualisiert werden.
Die Gestaltung eines komplexen Gebäudegrundrisses stellt in der Praxis aber sicherlich eine interessante Aufgabe dar, zu deren Lösung die Entscheidungskriterien erst noch erarbeitet werden müssen.

Die praktische Bewährungsprobe der Raumkomponente steht noch aus.

Literaturverzeichnis

[Allimant99]

Franck Allimant. The WinHelp Java Tutorial. WinHelp-Version der original HTML-Dokumentation auf der JavaSoft Website. 1999.
http://java.sun.com

[Apcon JWAM o.J.]

Apcon WPS. JWAM Framework: Architecture. Hamburg. o.J.
http://www.jwam.de/home/architecture.html

[Apcon T&M o.J.]

Apcon WPS. Tools & Materials Approach to Software-Development. Hamburg. o.J.
http://www.jwam.de/home/tm_approach.html

[Austeda89]

Franz Austeda. Lexikon der Philosophie. 6., völlig neubearbeitete Auflage. Wien. 1989.

[BannonSchmidt89]

Liam J. Bannon, Kjeld Schmidt. CSCW: Four Characters in Search of a Context. In: Proceedings of ECSCW'89. Gatwick. Großbritannien. 1989.

[Bäumer98]

Dirk Bäumer. Softwarearchitekturen für die rahmenwerkbasierte Konstruktion großer Anwendungssysteme. Dissertation. Universität Hamburg. 1998.

[Beck99]

Kent Beck. Extreme Programming explained: embrace change. Reading, Massachusetts. 1999.

[Bentley97]

Richard Bentley et.al. Basic Support for Cooperative Work on the World Wide Web. In: International Journal of Human Computer Studies: Special issue on Novel Applications of the WWW. Cambridge, Massachusetts. USA. 1997.

[Berlage99]

Thomas Berlage, Markus Sohlenkamp. Visualizing Common Artefacts to Support Awareness in Computer-Mediated Cooperation. In: K. Schmidt, et al. (Hrsg.). Computer Supported Cooperative Work: The Journal of Collaborative Computing. Vol.8, No.3. Dordrecht. Niederlande. 1999. (S.207-238)

[Beis00]

Christian Beis. Integration von Zugriffskontrollen in das JWAM-Framework. Diplomarbeit. Universität Hamburg. 2000.

[Bleeck97]

Wolf-Gideon Bleek. Techniken zur Konstruktion verteilter und technisch eingebetteter Anwendungssysteme. Diplomarbeit. Universität Hamburg. 1997.

[Börsenverein00]

Börsenverein des deutschen Buchhandels. Buch und Buchhandel in Zahlen 2000. Frankfurt a.M. 2000.
Auszug daraus unter: http://www.boersenverein.de/veroeff.htm

[Breitling00]

Holger Breitling. Unterstützung kooperativer Arbeitsprozesse: Vorgangsmappen im JWAM-Framework. Laufende Diplomarbeit. Universität Hamburg. 2000.

[Cotter95]

Sean Cotter, M. Potel. Inside Taligent Technology. Taligent, Inc. Reading, Massachusetts. USA. 1995.

[Duden90 5]

DUDEN Band 5. Fremdwörterbuch. 5. Auflage. Mannheim. 1990.

[Duden89 7]

DUDEN Band 7. Herkunftswörterbuch. 2. Auflage. Mannheim. 1989.

[Flanagan98]

David Flanagan. Java in a nutshell. 2. Auflage. Köln. 1998.

[Freund00]

Mirko Freund. Komponenten zur Kooperationsunterstützung: Das Postversandsystem. Diplomarbeit. Universität Hamburg. 2000.

[Fricke98]

Niels Fricke, Carola Lilienthal, Martin Lippert, Stefan Roock, Henning Wolf. Asynchrone Nachrichtenvermittlung für interaktive, kooperative Anwendungssysteme mit Java. Universität Hamburg. 1998.

[Gamma97]

Erich Gamma, et al. Entwurfsmuster: Elemente wiederverwendbarer Software. Aus dem Amerikanischen von Dirk Riehle. 1., korrigierter Nachdruck. Bonn. 1997.

[Gehrke98]

Achim Gehrke. Raum-Metaphern und Orientierungsunterstützung am Beispiel einer NetzEditorUmgebung nach der Werkzeug-Material- und der Gebäude-Metapher. Diplomarbeit. Universität Hamburg. 1998.

[GreenbergRoseman98]

Saul Greenberg, Mark Roseman. Using a Room Metaphor to Ease Transistions in Groupware. Research Report 98/611/02. Department of Computer Science, University of Calgary. Kanada. 1998.

[Greif88]

Irene Greif (Hrsg.). Computer-Supported Cooperatice Work: A Book of Readings. San Mateo, California. USA. 1988.

[Grudin94]

Jonathan Grudin. CSCW: History and Focus. University of California, Irvine. USA. 1994.

[Gryczan95]

Guido Gryczan. Prozeßmuster zur Unterstützung kooperativer Tätigkeit. Dissertation. Universität Hamburg. 1995.

[GutwinGreenberg99]

Carl Gutwin, Saul Greenberg. A Framework of Awareness for Small Groups in Shared-Workspace Groupware. Technical Report 99-1. Department of Computer Science, University of Saskatchewan. Kanada. 1999.

[HarrisonDourish96]

Steve Harrison, Paul Dourish. Re-Place-ing Space: The Roles of Place and Space in Collaborative Systems. In: Proceedings of CSCW'96. Cambridge, Massachusetts. USA. 1996. (S.67-76)

[JWAM99 1]

Guido Gryczan, Carola Lilienthal, Martin Lippert, Stefan Roock, Henning Wolf, Heinz Züllighoven. Frameworkbasierte Anwendungsentwicklung (Teil 1). In: OBJEKTspektrum 1/1999. (S.90-99)

[JWAM00 5]

Guido Gryczan, Andreas Havenstein, Stefan Roock, Ingrid Wetzel, Heinz Züllighoven. Frameworkbasierte Anwendungsentwicklung (Teil 5): Unterstützung von Kooperation mit persistenten fachlichen Behältern. In: OBJEKTspektrum 1/2000. (S.82-87)

[JWAM o.J.]

JWAM Code Styleguide. Universität Hamburg. o.O. o.J.

[Kilberth94]

Klaus Kilberth, Guido Gryczan, Heinz Züllighoven. Objektorientierte Anwendungsentwicklung: Konzepte, Strategien, Erfahrungen. 2., verbesserte Auflage. Wiesbaden. 1994.

[KrabbelWetzel00]

Anita Krabbel, Ingrid Wetzel. Designing Hospital Information Systems: Handling Complexity via a User-Oriented Document-Based Approach. In: Adi Armoni. Healthcare Information-Systems: Challenges of the New Millennium. London. 2000.

[Koch99]

Jörn Koch. Softwareprojekt HIPPO: Erfahrungsbericht zur Entwicklung des Fachlichen Modells nach WAM für einen Softwareprototyp zur Pflegeplanung- und Dokumentation. Studienarbeit. Universität Hamburg. 1999.

[Kuhn95]

Werner Kuhn. 7±2 Questions and Answers About Metaphors for GIS User Interfaces. In: T.L. Nyerges. et al. (Hrsg.). Cognitive Aspects of Human-Computer Interaction for Geografic Information Systems. Series D: Behavioural and Social Sciences. Vol.83. Dordrecht. Niederlande. 1995. (S.113-122)

[Kuhn96]

Werner Kuhn. Handling Data Spatially: Spatializating User Interfaces. Department of Geoinformation, TU Wien. 1996.

[Lippert99]

Martin Lippert. Die Desktop-Metapher in Systemen nach dem Werkzeug- und Material-Ansatz. Diplomarbeit. Universität Hamburg. 1999

[Mansfield97]

Tim Mansfield, et al. Evolving Orbit: a progress report on building locales. In: Proceedings of the Group'97 International Conference on Supporting Group Work. Phoenix, Arizona. USA. 1997. (S.241-250)

[Mansfield98]

Tim Mansfield, et al. Toward Locales: Supporting collaboration with Orbit. The University of Queensland. Australien. 1998.

[Mark97]

Gloria Mark, Ludwin Fuchs, Markus Sohlenkamp. Supporting Groupware Conventions through Contextual Awareness. GMD-FIT, German National Research Center for Information Technology. In: Proceedings of ECSCW'97. Dordrecht. Niederlande. 1997. (S.253-268)

[Mayer94]

Erwin Mayer. Synchronisation in kooperativen Systemen. Braunschweig. 1994.

[Oberquelle91]

Horst Oberquelle (Hrsg.). Kooperative Arbeit und Computerunterstützung - Stand und Perspektiven. Stuttgart. 1991.

[Oberquelle98]

Horst Oberquelle. Computergestützte kooperative Arbeit. Vorlesungsskript. Universität Hamburg. WiSe 1998/99.

[Rodden96]

Tom Rodden. Populating the Application: A Model of Awareness for Cooperative Applications. In: Proceedings of CSCW'96. Cambridge, Massachusetts. USA. 1996. (S.87-96)

[Röhring92]

Hans-Helmut Röhring. Wie ein Buch entsteht: Einführung in den modernen Buchverlag. 5. Auflage. Darmstadt. 1992.

[Rolf98]

Arno Rolf. Grundlagen der Organisations- und Wirtschaftsinformatik. Berlin. 1998.

[RoockWolf98]

Stefan Roock, Henning Wolf. Die Raummetapher zur Entwicklung kooperationsunterstützender Softwaresysteme für Organisationen. Diplomarbeit. Universität Hamburg. 1998.

[Schmid97]

Hans Albrecht Schmid. Systematic framework design by generalization. In: Communications of the ACM. Vol. 40. No.10. o.O. 1997. (S.48-51)

[Schülerduden85]

Schülerduden. Die Philosophie. Ein Sachlexikon der Philosophie. Mannheim. 1985.

[Shneiderman98]

Ben Shneiderman. Designing the User Interface. Strategies for Effective Human-Computer Interaction. 3rd Edition. Reading, Massachusetts. 1998.

[SimonSteinbrückner00]

Frank Simon, Frank Steinbrückner. Metrikbasierte Qualitätseinschätzung und Anomaliedetektion für das Softwaresystem Jwam 1.5. Universität Cottbus. 2000.

[Sohlenkamp98]

Markus Sohlenkamp, Wolfgang Prinz, Ludwin Fuchs. PoliAwaC – Design und Evaluation des PoliTeam Awareness Clients. In: Th. Herrmann, K. Just-Hahn. Groupware und organisatorische Innovation. Tagungsband D-CSCW'98. Stuttgart. 1998. (S.139-151)

[Standop90]

Ewald Standop. Die Form der wissenschaftlichen Arbeit. 13. Auflage. Heidelberg. 1990.

[Szyperski99]

Clemens Szyperski. Component Software: Beyond Object-Oriented Programming. London. 1999.

[Teufel95]

Stephanie Teufel, C.Sauter, T.Mühlherr, K.Bauknecht. Computerunterstützung für die Gruppenarbeit. Bonn. 1995.

[UML99]

Grady Booch, Jim Rumbaugh, Ivar Jacobson: Das UML-Benutzerhandbuch. Bonn. 1999.

[WAM98]

Heinz Züllighoven, et al. Das objektorientierte Konstruktionshandbuch nach dem Werkzeug & Material Ansatz. Universität Hamburg. 1998.

! Die in der Arbeit angegebenen Seitenzahlen beziehen sich auf die Fassung als .pdf-Datei, die dem Buch auf CD beiliegt.

[Wilkens00]

Mareile Wilkens. Analyse und Design für die Kooperationsunterstützung am Beispiel einer Krankenhausstation. Diplomarbeit. Universität Hamburg. 2000.

Verzeichnisse

Abbildungen

Abbildung 1: Das Benutzungsmodell im Spannungsfeld der Anforderungen 17
Abbildung 2: Das *three-layer model* der Orbit-Prototypen 29
Abbildung 3: Der Orbit-Mercury-Prototyp, Details 30
Abbildung 4: Das TeamWave-System, Werkzeuge 33
Abbildung 5: Der POLITeam Awareness Client 36
Abbildung 6: Die Schichten des JWAM-Frameworks 49
Abbildung 7: Package-Struktur des JWAM-Frameworks (Ausschnitt) 53
Abbildung 8: Kooperationsbild "Mailing-Aktion" 59
Abbildung 9: Kooperationsbild "Anfrage annehmen" 60
Abbildung 10: Der virtuelle Raum, Umgebungen, Plätze 70
Abbildung 11: Personen, Plätze und Ressourcen in den Beispiel-Szenarios 76
Abbildung 12: CRC-Karte "Einzelarbeitsplatz" 82
Abbildung 13: CRC-Karte "Raum" 84
Abbildung 14: CRC-Karte "Gebäude" 87
Abbildung 15: Visualisierung der Plätze 90
Abbildung 16: Fokussieren eines Raumes 94
Abbildung 17: Scrollen 95
Abbildung 18: "Zoomen" in Raumansicht 96
Abbildung 19: Reservieren durch Markieren 97
Abbildung 20: Awareness 98
Abbildung 21: Bewegen von Dingen 101
Abbildung 22: Bewegen von Inventar 103
Abbildung 23: Die Platz-Hierarchie der gemeinsamen Umgebung 109
Abbildung 24: Vision zur direkten Manipulation eines virtuellen Stapels 110
Abbildung 25: Vision zu abschließbaren Räumen 111
Abbildung 26: Vision zur Visualisierung von Etagen 112
Abbildung 27: Vision zum Chat-Tool in der Raumansicht 114
Abbildung 28: Vision zum Gebäudeplan-Menü 115
Abbildung 29: Vision zum Web-Interface 116
Abbildung 30: Gedachtes Raster auf der Fußbodenfläche eines Raumes 124
Abbildung 31: Gedachtes Raster im Raumplan eines Gebäudes 128
Abbildung 32: Gemeinsame und benutzerbezogene Komponenten des RoomExplorers 129
Abbildung 33: Client-, Server- und lokale Werkzeugkomponenten des RoomExplorers 130
Abbildung 34: RoomExplorerClient, RoomExplorer und RoomTop 143
Abbildung 35: Interaktionsdiagramm zum Start der lokalen Werkzeugkomponente 144

Tabellen

Tabelle 1: Zusammenhänge zwischen fachlichem und technischem Modell 16
Tabelle 2: Eigenschaften von Wert und Objekt 22
Tabelle 3: Konzepte zum Problem "Koordination über Material und Platz" 64
Tabelle 4: Konzepte zum Problem "gemeinsame Nutzung knapper Ressourcen" 65
Tabelle 5: Konzepte zum Problem "Handlungssicherheit" 67

Begriffe

Begriff 1: Szenario 13
Begriff 2: Kooperationsbild 14
Begriff 3: Benutzungsmodell 16
Begriff 4: Werkzeug 19
Begriff 5: Material 20
Begriff 6: Automat 20
Begriff 7: Behälter 21
Begriff 8: Kooperative Arbeit 23
Begriff 9: Koordination 23
Begriff 10: Kooperationsmittel 24
Begriff 11: Kooperationsmedium 25
Begriff 12: Awareness 27
Begriff 13: Ort 41
Begriff 14: Platz 42
Begriff 15: Framework 46
Begriff 16: Komponente 46
Begriff 17: Arbeitsumgebung 69
Begriff 18: Arbeitsplatz 69
Begriff 19: persönlich 71
Begriff 20: privat 71
Begriff 21: Gruppenwerkzeug 106

Schnittstellen

Schnittstelle 1: Room – Ausschnitt: Container-Schnittstelle 122
Schnittstelle 2: Room – Ausschnitt: Inventar-Schnittstelle 123
Schnittstelle 3: Room – Ausschnitt: Sondieren der Fußbodenfläche 124
Schnittstelle 4: Room – Ausschnitt: Positionieren von Dingen auf der Fußbodenfläche 125
Schnittstelle 5: Room – Ausschnitt: Beschreibungen eines Raumes 125
Schnittstelle 6: Building – Ausschnitt: Erweiterungen gegenüber Thing 127
Schnittstelle 7: dvFloorPlan – Ausschnitt: Erweiterungen gegenüber dvTableOfContents 127
Schnittstelle 8: BuildingImpl – Ausschnitt: Konstruktion eines konkreten Gebäudes 128
Schnittstelle 9: RoomExplorerServer – Ausschnitt: Manipulieren des Gebäudes 133
Schnittstelle 10: RoomExplorerServer – Ausschnitt: Benutzereinstellungen 134
Schnittstelle 11: RoomExplorerServer – Ausschnitt: Sondieren des Schnappschusses 135
Schnittstelle 12: RoomExplorerSnapshot – Ausschnitt: Sondieren des Serverzustands 137
Schnittstelle 13: RoomExplorerSnapshot – Ausschnitt: Handhaben von Änderungen 138
Schnittstelle 14: RoomExplorerClient – Ausschnitt: "tryTo"- und "canTryTo"-Operationen 140
Schnittstelle 15: RoomExplorerClient – Ausschnitt: Ermitteln des "gepufferten" Schnappschusses und der Events zu dessen Änderungen 141

Anhang

<<Waschzettel-Abbildung nicht in PDF-Version>>